One-shot Face Reenactment

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab
CAIM Computer-assisted Applications in Medicine

One-shot Face Reenactment

Master's Thesis

Siwei Zhang
Department of Information Technology and Electrical Engineering

Advisor: Dr. Tiziano Portenier, Cheng Li
Supervisor: Prof. Dr. Orcun Goksel

September 15, 2019

Abstract

To enable realistic shape (e.g. pose and expression) transfer, existing face reenactment methods rely on a set of target faces for learning subject-specific traits. However, in real-world scenario end-users often only have one target face at hand, rendering existing methods inapplicable. In this work, we bridge this gap by proposing a novel one-shot face reenactment learning framework. Our key insight is that the one-shot learner should be able to disentangle and compose appearance and shape information for effective modeling. Specifically, the target face appearance and the source face shape are first projected into latent spaces with their corresponding encoders. Then these two latent spaces are associated by learning a shared decoder that aggregates multi-level features to produce the final reenactment results. To further improve the synthesizing quality on mustache and hair regions, we additionally propose FusionNet which combines the strengths of our learned decoder and the traditional warping method. Extensive experiments show that our one-shot face reenactment system achieves superior transfer fidelity as well as identity preserving capability than alternatives. More remarkably, our approach trained with only one target image per subject achieves competitive results to those using a set of target images, demonstrating the practical merit of this work.

Acknowledgements

First and foremost, I would like to address my sincere gratitude to Dr. Tiziano Portenier and Prof. Dr. Orcun Goksel for their supervision and continuous support. Their previous and insightful advice has offered me a grand new perspective and understanding for this research topic. They have also spent much time reading through the draft patiently and provided me helpful suggestions for my writing.

I would also like to thank Cheng Li, my co-supervisor in SenseTime Research, who provided me the opportunity to implement this thesis in the company. And special thanks to Yunxuan Zhang, Yue He, and Ziwei Liu, my colleagues in SenseTime Research, for their support. We have worked as a great team to finally bring this research project into a publication.

Finally, my deepest gratitude goes to my beloved parents, who have provided the strongest support for me all along the way, and have always encouraged me to pursue my dream.

Contents

List of Figures

Chapter 1

Introduction

1.1 Focus of this Work

Face reenactment is an emerging conditional face synthesis task that aims at fulfilling two goals simultaneously: 1) transfer a source face shape (facial expression and pose) to a target face; 2) preserve the appearance and the identity of the target face. In recent years, it has attracted enormous research efforts due to its practical values in virtual reality and entertainment industries.

An ideal face reenactment system should be capable of generating a photo-realistic face sequence following the pose and expression from the source sequence when only **one shot or few shots** of the target face are available. However, to enable realistic shape (such as pose and expression) transfer, existing face reenactment methods rely on a set of target faces for learning subject-specific traits. Some recent works [57, 25, 9] relaxed the problem by assuming that the reenactment system can be trained on a relatively long video from the target face. For each target face, they train a subject-specific generator using an auto-encoder framework. All these methods require a long video from the target and long training time to obtain the target-specific model, limiting their potential applications.

Another line of work explores few-shot synthesis [6, 7, 67, 31, 64], but either need paired data for training [64], or cannot deal with large changes in head pose [67], with unsatisfying performance.

In this work, we propose a novel one-shot face reenactment learning framework to bridge this gap. In our problem setting, we assume that only a single shot is available for each person no matter during training or testing. Several challenges exist for one-shot face reenactment: 1) The appearance of the target person is partial for all views since we only have one reference image from the target person. Synthesizing an image with an arbitrary view with such a limited input constraint is still an open question. 2) A single image can only cover one kind of expression. The appearance of other expressions (closed eyes, opened mouth) is not available, making the training more challenging. 3) Identity preserving when only a single target shot is available becomes a hard problem.

We believe the key to overcoming the aforementioned challenges is the capability of disentangling and composing appearance and shape information. To this end, the proposed one-shot learner first disentangles the appearance information $a \in \mathcal{A}$ provided by the target face and shape $b \in \mathcal{B}$ provided by source. Then it combines this information by a decoder $D : \mathcal{B} \times \mathcal{A} \rightarrow \mathcal{X}$, which takes full advantage of a so that the model can be trained by one-shot data, where the shape information b is represented by face parsing maps. The disentangled feature space is visualized in Figure 1.1.

The identity representation a is critical here. We expect the representation keeps as many facial texture details as possible, but not pose and expression dependent. Achieving this goal is nontrivial: 1) Face representation learned by some difficult tasks like face verification seems to be a good candidate. However, our experiments show that it cannot retain low-level texture information well. 2) We require a

representation that can support face reconstruction but exclude the facial pose and expression of the input. 3) There exists a natural gap between training and testing. During the testing phase of the reenactment, the appearance a and shape b belong to different identities. However, it is infeasible to obtain face pairs with the same head pose and expression across different identities for training.

We found that inappropriate encoding of appearance can hurt model performance considerably, for example, losing facial details such as eye colors or lip thickness due to insufficient information carried by the encoder's output, or identity shift due to facial structure changes. To supervise our shape-independent face appearance representation, we link two branches over the face appearance decoder: one for appearance reconstruction and the other one for the reenactment. The two branches are trained jointly. These two branches both take the output of the appearance encoder as input. In other words, an auto-encoder is trained for appearance encoding and reconstruction, and a reenactment decoder is trained in parallel with the appearance decoder to transfer pose to the target face.

To alleviate appearance and identity information loss during the reenactment process, we propose a novel strategy that inserts feature maps of different scales from the appearance decoder into multiple intermediate layers of the reenactment decoder to make full use of the appearance information. Besides, we preserve shape semantic information by inserting SPatially-Adaptive (DE)normalization (SPADE) modules [43] into the reenactment decoder in a multi-scale manner to achieve a semantically adaptive decoder. Additionally, we propose a FusionNet to improve the performance on mustache and facial texture generation.

The contributions of this work can be summarized as follows: 1) We study a new problem "one-shot face reenactment". This study offers a practical framework for real reenactment scenarios with the notion of one-shot training and test settings. 2) We propose a novel one-shot face reenactment learning framework. It disentangles and composes appearance and shape information for effective modeling. We additionally design FusionNet that combines the strengths of synthesis-based method and warping-based approach to better preserve texture information. Extensive experiments validate our framework that it can be trained with one-shot data while still able to reenact realistic face sequences with just only one image from the target person. More remarkably, the results of the proposed approach outperform state-of-the-art single-image face generating methods and are even competitive with subject-specific methods.

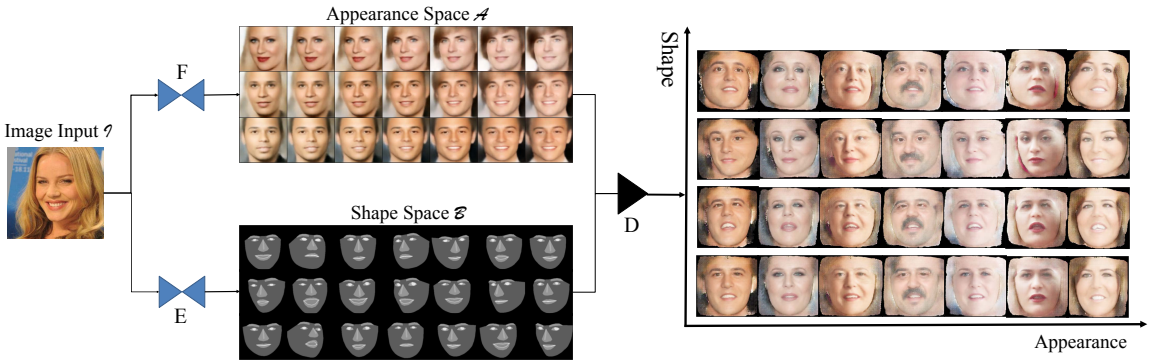


Figure 1.1: **Disentangled feature space:** Two separate encoders are adopted to disentangle the appearance and shape spaces, and a joint reenactment decoder composes encoded features into reenactment results. Here the appearance space visualization is reconstructed reference images, and the shape encoder outputs face parsing maps.

1.2 Thesis Organization

The thesis is organized into the following chapters. Chapter 2 gives a brief overview of related work, including generative models, the development of variations of the Generative Adversarial Nets (GAN), image-to-image translation methods, face attribute editing methods, pose transfer methods, and face reenactment methods. Chapter 3 describes the problem formulation, model design motivation, the model architecture and the objective function in detail, including the post-processing module FusionNet. Chapter 4 explains the experiment setting, evaluation metrics, and the experiment results of several baselines and our proposed model. Qualitative and quantitative comparisons are implemented between our proposed model and baselines. An ablation study is implemented to investigate the effect of the proposed multi-scale appearance feature concatenation on identity consistency. Chapter 5 discusses the failure cases and gives conclusions of the work.

1.3 Terminalogy

Here some specific terminalogy is introduced as following:

- Conv: Convolution
- Deconv: Deconvolution
- BatchNorm: Batch Normalization [20], which normalizes the samples in one mini-batch in the channelwise-manner to handle the internal covariate shift problem.
- InstanceNorm: Instance Normalization [53], which performs normalization for each sample independently in the channelwise-manner.
- ReLU: Rectified Linear Unit [42], an activation function $f(x) = max(0, x)$.
- LeakyReLU: Leaky Rectified linear Unit [38], an activation function as described below:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \lambda x & \text{if } x \leq 0 \end{cases} \tag{1.1}$$

Chapter 2

Related Work

This thesis mainly focuses on facial image synthesis, therefore the related work will give an overview in generative models, image-to-image translation, face attribute editing, pose transfer and face reenactment.

2.1 Generative Models

The most basic goal of generative models is to learn a mapping from noise into complex real data distributions. There are two main generative models for high-quality image synthesis, namely Auto-Encoding Variational Bayes (VAE) [27] and Generative Adversarial Network (GAN) [17].

2.1.1 Variational Auto-encoder (VAE)

Kingma *et al.* [27] proposes VAE as a stochastic variational inference and learning algorithm to deal with cases where latent variables have intractable posterior distributions, by reparameterization of the variational lower bound. The algorithm brings a differentiable and unbiased estimator of the lower bound, which can be simply optimized by the standard stochastic gradient descent method. A neural network similar as the auto-encoder is implemented. The problem formulation is explained as following: for a dataset $X = \{x_i\}_{i=1}^{N}$ which includes N i.i.d. samples of variable x generated by some random process, involving a latent variable z which cannot be observed, the process to generate a value x in dataset X is assumed to be divided into two steps: first to generate a z from a prior distribution $p_\theta(z)$, then to generate an x from a conditional distribution $p_\theta(x|z)$, therefore the real generative model can be represented by $p_\theta(z)p_\theta(x|z)$.

However the marginal likelihood $p_\theta(z|x) = \int p_\theta(z)p_\theta(x|z)dz$ and the true posterior $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x)$ are intractable, which makes it difficult to learn by maximizing the marginal likelihood. To deal with this problem, VAE introduces a new recognition model $q_\phi(z|x)$ as an approximation of the true posterior $p_\theta(z|x)$, by learning the recognition model and the generative model jointly, as illustrated in Figure 2.1, which can transform the problem of maximizing the marginal likelihood into maximizing a lower bound of the marginal likelihood.

In the commonly used implementation of VAE the neural network is adopted, where the recognition model $q_\phi(z|x)$ is referred as the probabilistic encoder, and the $p_\theta(x|z)$ is referred as the probabilistic decoder, which can be seen as a conditional generative model. The encoder takes a random sampled z from a prior distribution of latent variable z referred as $p_\theta(z)$ to learn the parameters ϕ of $q_\phi(z|x)$. The prior distribution is usually assumed to be a centered isotropic multivariate Gaussian distribution $p_\theta(z) = N(Z; 0, I)$. In the task for image synthesis, it is usually assumed that the posterior $q_\phi(z|x)$ is a

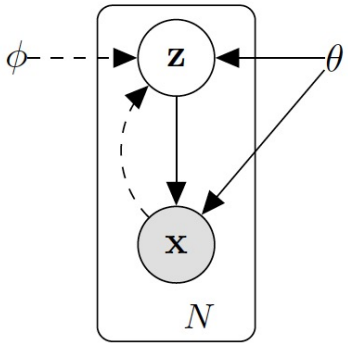


Figure 2.1: **VAE model assumption**: a recognition model $q_\phi(z|x)$ (denoted by the dashed lines) is utilized to approximate the true posterior $p_\theta(z|x)$ (where the generative model is denoted by the solid lines), and the model parameters ϕ and $theta$ are learnt jointly. Figure taken from [27].

a multivariate Gaussian with a diagonal covariance structure:

$$q_\phi(z|x) = N(z; \mu, \sigma^2 I) \tag{2.1}$$

Therefore the mean and standard deviation μ, σ^2 are the output of the encoder part. Then we randomly sample a vector z^l from $q_\phi(z|x) = N(z; \mu, \sigma^2 I)$:

$$z^l = \mu + \sigma\epsilon \quad \text{with} \quad \epsilon \in N(0, 1) \tag{2.2}$$

as the input of the decoder $p_\theta(x|z)$ to generate the data point x. The encoder and decoder of VAE can be both implemented by Multilayer Perceptron (MLP) or Convolutional Neural Networks (CNN), i.e., nonlinear functions. For training of VAE, a Kullback-Leibler divergence (KL divergence) loss and a data reconstruction loss are optimized jointly:

$$L = -D_{KL}((q\phi(z)||p\theta(z)) + L_x = \frac{1}{2}\sum_j (1 + log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) + \|x - x'\| \tag{2.3}$$

Where x' is the reconstructed data point, i.e., the output of VAE. In the generative phase, a random z_j is directly sampled from learned variational posterior distribution $q_\phi(z|x)$ and fed to the decoder for generation. Due to the sampling procedure of z_j, the synthesized images of VAE can sometimes be blurry.

2.1.2 Generative Adversarial Network (GAN)

Goodfellow *et al.* [17] proposes the Generative Adversarial Network (GAN), which has become a popular method in recent image synthesis research, and has developed into many different variations for better learning performance. Different from VAE which learns by approximating the posterior distribution of a latent variable and sampling, GAN directly learns the data distribution in an adversarial way by iterative training of a generator and a discriminator. The generator aims to generate images of high quality to fool the discriminator, while the discriminator aims to distinguish the generated fake images from the real data. This section includes a brief overview of the vanilla GAN and several variations of later GANs.

GAN Goodfellow *et al.* [17] applies the adversarial training incorporating a generator and a discriminator. The discriminator learns to distinguish whether a sample is from the true distribution or a fake sample generated by the generator, and the generator tries to generate fake samples which are as similar as the true data as possible to fool the discriminator. The input to the generator is a random noise sample z from the distribution of $p_z(z) \sim N(0, 1)$. The iterative training of the generator and discriminator will hopefully reach an equilibrium when the generator can generate a data sample which is indistinguishable from the true data, and the discriminator cannot distinguish the true data and generated sample anymore. The vanilla GAN is implemented by MLP in both the generator and the discriminator.

Let $G(z; \theta_g)$ denote the generator, and $D(x; \theta_d)$ denote the discriminator, with θ_g, θ_d representing the parameters for G and D respectively. Therefore $G(z; \theta_g)$ will try to map the noise input z into the true data distribution, and $D(x; \theta_d)$ takes fake data or real data as the input to predict a probability for the input sample to come from the true distribution. GAN plays a two-player minimax game as following:

$$min_G max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[log(D(X))] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))] \tag{2.4}$$

To be specific, during each two consecutive training steps, in the first step, with G fixed, D is trained to maximize the probability of predicting the current label (fake or real) to both samples generated by G and from true data. Then in the second step G is trained to minimize $log(1 - D(G(z))$ with D fixed. Figure 2.2 shows an overview of GAN training process.

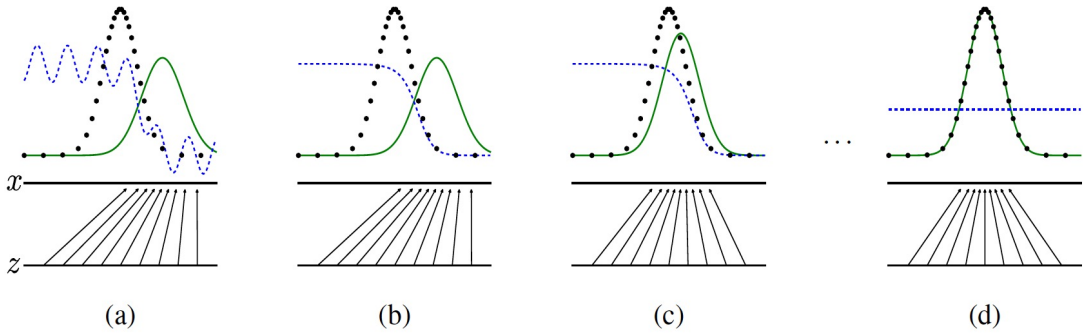


Figure 2.2: **The training process of GAN**: GAN learns to map the random sample z into a realistic data point x, which is indistinguishable from the true data distribution (the black dots). The solid green curve denotes the data distribution generated by the generator, and the blue dashed curve denotes the discriminative distribution. (a) represents a training stage near convergence, then D is optimized for one step, and D^* is reached in this step in (b). (c) shows the training step of G, when gradients from D optimize G to the true distribution, and the final global optimum (d) is reached after iterations of (b) and (c) in an ideal situation, when the generated distribution equals to the true data distribution. Figure taken from [17].

The global optimum for this minimax game is $p_g = p_{data}$, i.e., the generator will successfully learn to map a random noise to the true distribution when the global optimum is reached. However, the vanilla GAN training can be unstable sometimes, and the MLP layers are not capable to generate high-quality images. The following GAN variations are all based on the principle of two-player minimax iterative training.

DCGAN Radford *et al.* [46] proposes the Deep Convolution Generative Adversarial Networks (DCGAN) to build the GAN network upon the CNN architecture to reach a more stable training process

across datasets in several different image domains, such as human face or indoor scenes. An example for the generator architecture in DCGAN is illustrated in Figure 2.3, which takes a random sample z as input, and consists of several deconvolutional layers for image generation.

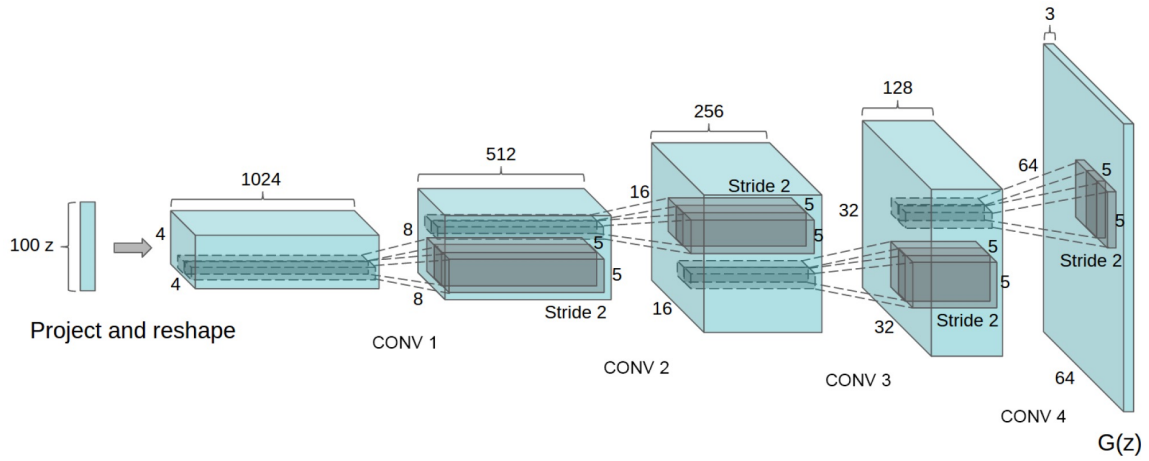


Figure 2.3: **Generator of DCGAN**: a fully convolutional architecture with consecutive deconvolutional layers to map a random noise into a realistic image. Figure taken from [46].

The guidelines to design a stable DCGAN includes the following:

- Replace pooling layers with convolutional layers.
- Discard all fully-connected layers.
- Use BatchNorm layers in both generator and discriminator.
- Use ReLU in the generator (except for the last layer before output, use Tanh instead) and LeakyReLU in the discriminator.
- Use Adam optimizer [26] with a learning rate of 0.0002.

By interpolation between different random samples of z, DCGAN can visualize a 'walk in the latent space' to slowly transform a generated image into another generated image, indicating that the model can learn semantic representations in the continuous feature manifold. In addition, the discriminator in DCGAN also achieves competitive performance in image classification task. DCGAN is still widely used in recent research due to its training stability and capacity for high-quality image synthesis. However, DCGAN only proposes a new base architecture for GAN training, but cannot solve the unstable training caused by the objective function in GAN.

WGAN In principal, the vanilla GAN and DCGAN both optimizes the Jensen-Shannon divergence (JS divergence) between the generated distribution P_1 and the true distribution P_2:

$$JS(P_1\|P_2) = \frac{1}{2}KL(P_1\|\frac{P_1+P_2}{2}) + \frac{1}{2}KL(P_2\|\frac{P_1+P_2}{2}) \tag{2.5}$$

However, the JS divergence will be a constant number in cases when the two distributions P_1 and P_2 are far away from each other without any intersection, which causes serious problems for training due to the

vanishing gradient, and also is the most fundamental reason for the unstable training of GAN. To deal with this, Arjovsky *et al.* [4] proposes Wassertein-GAN (WGAN) to replace the JS divergence in the objective function by the Wasserstein distance:

$$W(\mathbb{P}_1, \mathbb{P}_2) = \inf_{\gamma \in \Pi(\mathbb{P}_1, \mathbb{P}_2)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{2.6}$$

where $\Pi(\mathbb{P}_1, \mathbb{P}_2)$ represents all possible joint distributions combining P_1 and P_2. Comparing with KL divergence or JS divergence, Wasserstein distance still reflects the distance between P_1 and P_2 even they are far from each other. See Figure 2.4 for a comparison between WGAN and GAN training.

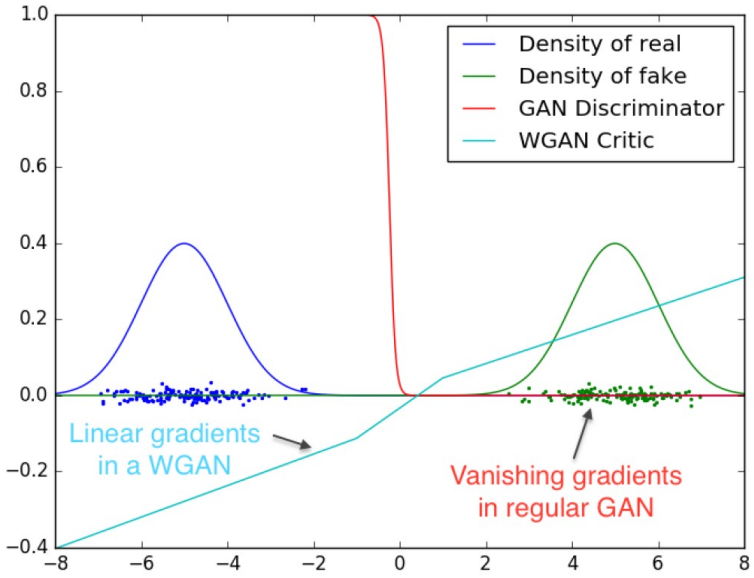


Figure 2.4: **Comparison between W-GAN and GAN training**: during the training process of regular GAN, the discriminator gradient will become zero if the generated distribution and true distribution are far from each other, which is caused by the JS divergence in GAN loss function. In the training of WGAN, as Wasserstein distance can still reflect the distance between two distributions when they are far from each other, the WGAN discriminator gradients are linear, which can greatly improve training stability. Figure taken from [4].

By using such Wasserstein distance in the loss function for training, WGAN can greatly improve the training stability, and solves the problem of mode collapse in GAN.

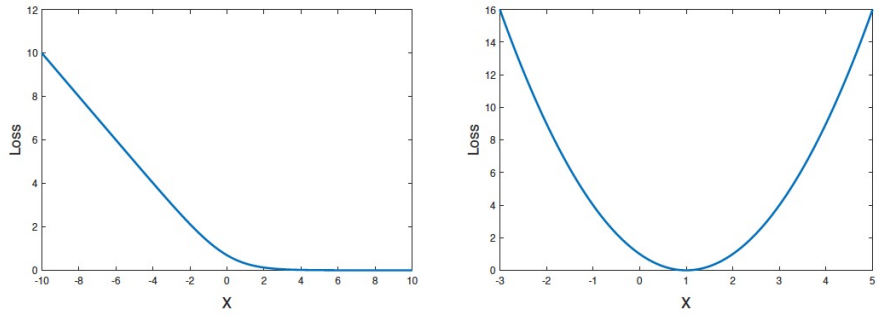
WGAN-GP To fulfill the Lipschitz restriction (magnitude of gradients of the discriminator must be smaller than a constant), the weights in WGAN are strictly truncated to a range by weight clipping, which may generate poor samples or fail to converge. An alternative to weight clipping is proposed in WGAN-GP [18] to enable a more stable training process. In particular, the gradient penalty (GP) is introduced to constrain the gradient norm of the critic's output with respect to its input, in order to enforce the Lipschitz restriction, as shown by the third term in the loss function of WGAN-GP:

$$L = \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[D(\boldsymbol{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[\left(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1\right)^2\right] \tag{2.7}$$

where $\mathbb{P}_r$ and $\mathbb{P}_g$ denote the true data distribution and generated distribution respectively, and $\hat{\boldsymbol{x}}$ is a random interpolated sample between pairs of points sampled from $\mathbb{P}_g$ and $\mathbb{P}_r$. Since it is intractable to enforce the gradient constraint everywhere, the constraint is only enforced on the interpolated samples.

LSGAN Mao *et al.* [39] proposes the Least Squares Generative Adversarial Networks (LSGAN) to improve GAN training process by modification of the loss function. Instead of the cross entropy loss of the discriminator in GAN, which utilizes the discriminator as a two-class classifier, LSGAN proposes to replace discriminator loss with a least square loss function.

As shown in Figure 2.5, the cross entropy loss in the discriminator will face vanishing gradient during training process, which results in unstable training. However, the least square loss adopted in LSGAN can solve this problem to improve training stability.



(a) (b)

Figure 2.5: **Comparison between LSGAN and GAN**: the cross entropy loss in the discriminator of GAN faces the problem of vanishing gradient (a) during training, therefore replaced by the least square loss (b) in LSGAN. Figure taken from [39].

The cross entropy loss can only distinguish the authenticity of the data, but produce small loss values for the fake samples if they are on the correct side of the decision boundary, resulting little information for updating the generator. By adopting the least square error for discriminator loss, even the fake sample is in the right side of the decision boundary, the least square loss will evaluate the distance between the fake sample and the decision boundary, therefore producing larger punishment for fake samples that are further from the decision boundary. The punishment for fake samples will encourage the generator push fake samples closer to the decision boundary, or the true distribution, as the decision boundary is supposed to go through the true data distribution after a successful training.

The loss function of LSGAN is defined as following:

$$\begin{aligned} \min_D V_{\mathrm{LSGAN}}(D) &= \tfrac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \tfrac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}(\boldsymbol{z})\left[(D(G(\boldsymbol{z}))-a)^2\right] \\ \min_G V_{\mathrm{LSGAN}}(G) &= \tfrac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right] \end{aligned} \tag{2.8}$$

where z is the input noise, x denotes the true data sample. G, D denote the generator and discriminator respectively. The selection of a, b, c should satisfy $b - c = 1$ and $b - a = 2$, or $b = c$, such as $a = 0, b = 1, c = 1$.

The generator takes a random sample as the input, which is fed through a fully-connected layer and consecutive [deconv, BatchNorm] layers. The discriminator consists of several convolutional layers and a fully-connected layer.

Conditional GAN Mirza *et al.* [41] proposes the Conditional Generative Adversarial Nets (Conditional GAN) as the conditional version of GAN, which aims to generate data conditioned on a certain requirement, for example, a class label, to allow control over model outputs. In the problem setting, besides the noise input z, the generator and the discriminator are both conditioned on an additional input y, which is fed to the generator and discriminator as the additional input layer. The loss function is defined as following:

$$min_G max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[log(D(x|y))] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z|y)))] \tag{2.9}$$

The model structure is illustrated in Figure 2.6.

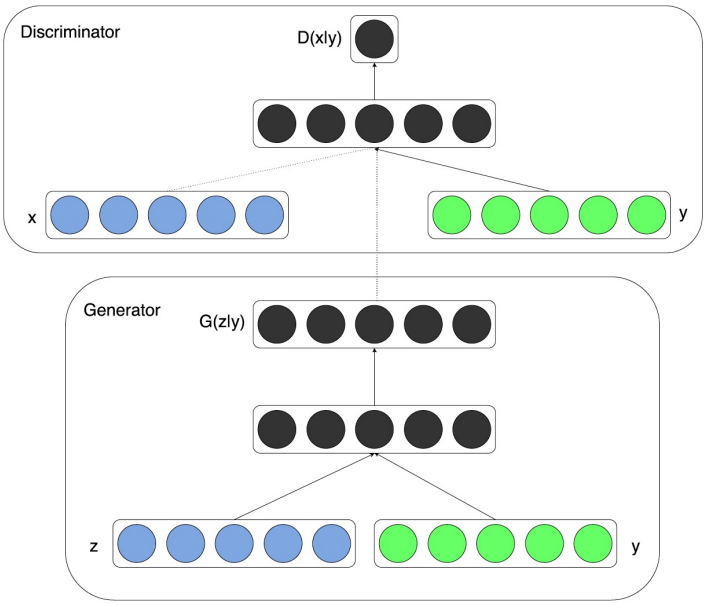


Figure 2.6: **Conditional GAN**: the generator takes the input noise z and the additional condition y as inputs. The discriminator also takes y as an additional input. Figure taken from [41].

PG-GAN Karras *et al.* [23] proposes Progressive-Growing GAN (PG-GAN) by a new training methodology for high-resolution image synthesis. The training starts from low resolution with few layers, and progressively add layers to increase fine details in the generated image. Figure 2.7 illustrates the progressive training process. This progressive growing training method can speed up and stabilize the training process. Comparing with directly generating high-quality images, it is much easier to first generate low resolution images with fewer modes, and adding layers progressively can decompose the difficult problem into many simpler questions step by step.

In addition, to avoid a sudden stroke for the training, PG-GAN fades in new layers smoothly each time when doubling the resolution. The new added layer will be treated as a residual block with a weight parameter of α, with a weight $1 - \alpha$ for the existing layer. The value of α is gradually increased from 0 to 1 to smooth the transition period.

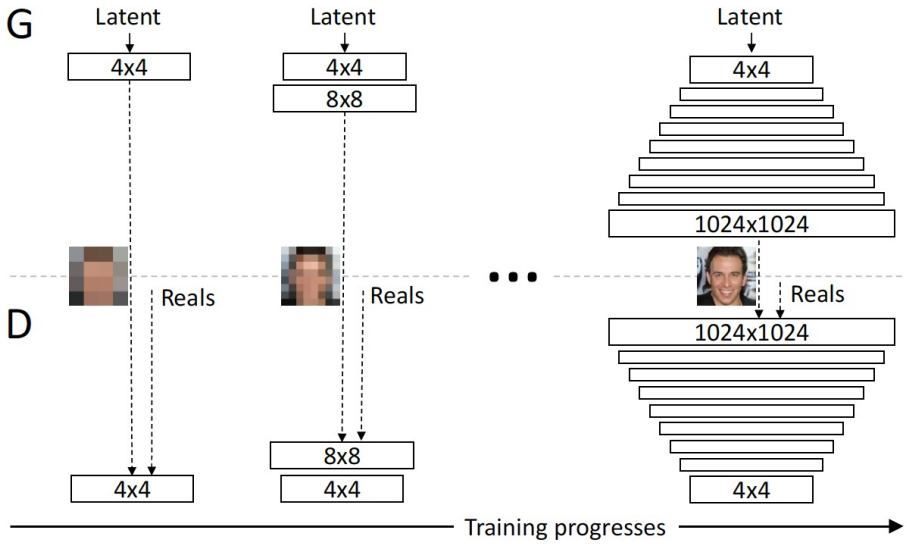


Figure 2.7: **Progressive training of GAN**: the training starts with a low resolution of 4x4 for both the generator and discriminator. Then layers are added progressively for finer details generation, and all the earlier layers remain trainable during the whole training stage. Figure taken from [23].

2.2 Image-to-image Translation

Image-to-image translation refers to the task to translate one possible representation of a scene into another, for example, to map a sketch image into a photo, or map a photo image into the cartoon style. The image-to-image translation methods can be roughly divided into two categories with respect to the training method: paired training and unpaired training.

Paired training Paired training requires paired image with the same content during the training stage, with pix2pix [21] and pix2pixHD [55] as two representative methods. Pix2pix implements a conditional GAN using a U-Net shape generator, with the adversarial loss and the reconstruction loss in the training objective function. Comparing with previous adopted encoder-decoder structure for the generator, the U-Net shape-like generator can better capture low-level information through the skip connections, which is essential for image-to-image translation problems. In addition, a patchGAN is proposed for the discriminator to generate images with clearer details and higher resolution. The vanilla GAN discriminator only produces a label for the image as fake or real, but ignores the local details. PatchGAN tries to classify if each NxN patch in an image is real or fake to model high-frequencies in the image.

Wang *et al.* [55] proposes pix2pixHD to generate images with higher resolution. As shown in Figure 2.8, the model adopts a coarse-to-fine pipeline. A global generator G_1 is firstly trained to generate image at half resolution, and then the local enhancer G_2 is added to train the generated image together with G_1 at full resolution of 2048x1024. Besides the combination of two generators of different resolutions, the multi-scale discriminator is proposed to enlarge the receptive field of the discriminator to better facilitate the synthesis. Three discriminators with identity structures but different image scales are incorporated to distinguish images from different resolutions. The one that works in the coarsest scale has the largest receptive field to decide the image authenticity from a global view, while other two discriminators encourage the generator to produce fine details. A special feature matching loss is added to the adversarial

loss upon the discriminator: features are extracted from multiple layers from the discriminator to match these features for real and fake images. The feature matching loss can stabilize the training as the generator will produce realistic images in multiple scales.

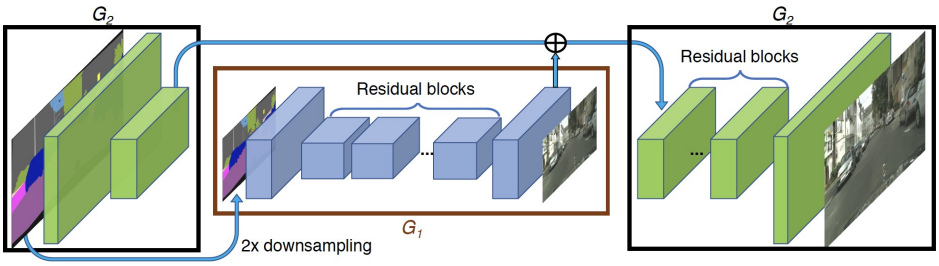


Figure 2.8: **Architecture of pix2pixHD**: a coarse-to-fine pipeline is adopted to generate high resolution images. A global generator G_1 is first trained to produce image at half resolution, and later the local enhancer G_2 is added to generate image at full resolution, trained jointly with G_1. Figure taken from [55].

Image synthesis guided by sketch, color and texture is implemented in TextureGAN [58], with input of a sketch and a texture patch in the sketch. Besides GAN, VAE is also widely used in image-to-image translation. Esser *et al.* [15] proposes the Variational U-Net (VU-Net) as a conditional U-Net for shape guided image generation, with a variational auto-encoder to model the appearance distribution. Some combines VAE and GAN for better generation quality [43, 68]. Recent work gauGAN [43] replaces the BatchNorm layers with spatially adaptive normalization layers to directly feed semantic information into multi-layers by SPADE modules, allowing user control over both semantic and style for image generation. Realistic-looking images in different views are generated in VariGANs [68] in a coarse-to-fine manner, by first generating the coarse image via performing variational inference to model the global appearance, and then forwarding the coarse image to adversarial training for fine details.

Unpaired training Paired training methods impose high requirements for training data, as paired images are usually difficult to obtain. Recent research works have proposed many image-to-image translation with unpaired images during training stage.

Zhu *et al.* [22] proposes CycleGAN to learn a mapping to translate images from one domain into another domain by unsupervised learning. Given two domains X and Y, training samples $\{x_i\}_{i=1}^N$ where $x_i \in X$ and $\{y_j\}_{j=1}^M$ where $y_j \in Y$, as the training samples are not paired, to get supervision information, the key idea is to implement two mapping functions G and F for a forward and backward translation where $G : X \rightarrow Y$ and $F : Y \rightarrow X$. Two discriminators D_X and D_Y are used for two domains respectively, with D_X aiming to distinguish between x and the generated image $F(y)$, and D_Y aiming to distinguish between y and the generated image $G(x)$. The adversarial loss of F, G are represented by:

$$\begin{aligned}\mathcal{L}_{GAN}(G, D_Y, X, Y) &= \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))] \\ \mathcal{L}_{GAN}(G, D_X, Y, X) &= \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log(1 - D_X(F(y))]\end{aligned} \tag{2.10}$$

With the cycle reconstruction process, the cycle consistency loss is defined as:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1] \tag{2.11}$$

The cycle reconstruction idea eliminates the requirement for paired training data, to provide supervision under the unsupervised setting, therefore CycleGAN has become the foundation for many later

image synthesis research works, including many face attribute editing methods [35, 66, 45, 10] which will be introduced in the next section.

2.3 Face Attribute Editing

Face attribute editing refers to editing certain face attributes while keep the remaining parts of the face unchanged, such as gender transformation, aging, expression editing, which can be considered as a special topic in image-to-image translation. Given the source domain and the target domain, the attribute editing can be implemented by the image-to-image translation methods, for example, CycleGAN [22]. There are mainly two different streams for face attribute editing algorithms, which are cycle-based, and disentanglement-based.

Cycle-based Cycle-based methods [35, 66, 45, 10] treat the face attribute editing problem as an image-to-image translation task, and utilize the cycle loss as proposed in CycleGAN to deal with lack of paired data.

Chang *et al.* [10] proposes pairedCycleGAN to implement style transfer to apply makeup for human faces given a reference person and the corresponding makeup style. Let X and Y denote the no-makeup domain and with-makeup domain, then only one image is available for each style. Let β denote a makeup style, then the input of the model will be a source image without makeup denoted by $x \in X$, and a reference image with makeup style β, denoted by $y^\beta \in Y$. The model involves two asymmetric functions: a forward function G to encode example-based style transfer, which takes the source image x and the reference style (an example photo y^β of a different person) as inputs, and a backward function F to remove the makeup, with the stylized image as the input to generate the corresponding person image without makeup. As illustrated in Figure 2.9, the training pipeline includes two stages.

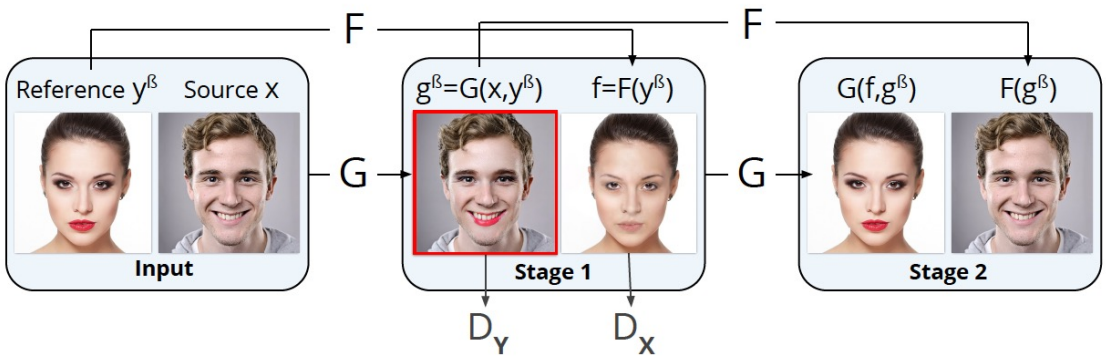


Figure 2.9: **pairedCycleGAN**: a forward function G takes the source image and the reference image as inputs to add makeup for the source image, and a backward function F aims to remove the makeup. Figure taken from [10].

During the first stage, the source image x is stylized by G, and F is applied to the reference image y^β to remove the makeup:

$$\begin{aligned} g^\beta &= G(x, y^\beta) \\ f &= F(y^\beta) \end{aligned} \tag{2.12}$$

During the second stage, the stylized source image is fed into F to remove the added makeup, while the reference face which is now free of makeup is fed to G to recover the makeup, which completes the cycle

transformation:

$$\begin{aligned} y_{reconstruct} &= G(f, g^{\beta}) \\ x_{reconstruct} &= F(g^{\beta}) \end{aligned} \tag{2.13}$$

The training loss includes three parts: the adversarial loss for G and F, the identity loss for x, and the style loss for y^{β}. To be specific, the identity loss is the L1 loss between the source image x and the reconstructed $x_{reconstruct}$, the style loss is the L1 loss between the reference image y^{β} and the reconstructed $y_{reconstruct}$, i.e., the cycle loss. The learned model can apply makeup and makeup removal given an arbitrary pair of source image and reference image.

Lu *et al.* [35] proposes a conditional CycleGAN to combine face attribute editing and super-resolution. The model takes a low-resolution image x, and a target attribute vector (e.g., blond hair) as inputs, and generate a high resolution image with the target attribute, which can be formulated as a problem to map domain X to domain Y subjected to an attribute condition Z. Two generators are incorporated, with one generator G_B taking low-resolution image and target attribute vector to generate high resolution image, and the other generator G_A taking the generated high resolution image to reconstruct the low resolution input. Two branches are included in training: in the first branch, a high resolution image is fed to G_A to generate low-resolution image, which is combined with attributes from the high resolution image (the target attributes) and then fed to G_B to reconstruct the high resolution image. In the second branch, a low-resolution image combined with the target attributes goes through G_B to generate a high resolution image, then fed to G_A to reconstruct the input low resolution image, with the cycle consistency loss, adversarial loss, and face verification loss included in the objective function.

Spatial attention mechanism is adopted in SaGAN [66] and GANimation [45]. SaGAN takes an image I and a target attribute value c as inputs for attribute editing. The generator includes an attribute manipulation network (AMN) and a spatial attention network (SAN), and the key idea is to only alter the attribute-specific region and keep the rest facial regions unchanged. The AMN focuses on how to manipulate while the SAN aims to locate where to modify. Let F_m denote the AMN, and F_a denote the SAN, then AMN tries to generate an edited face image:

$$I_a = F_m(I, c) \tag{2.14}$$

and SAN outputs an attention mask which focuses on the attribute-specific region:

$$b = F_a(I) \tag{2.15}$$

The final result is a combined result of I_a and the spatial attention mask b:

$$\hat{I} = I_a \cdot b + I \cdot (1 - b) \tag{2.16}$$

An example of the attention mask is shown in Figure 2.10, where we can see that the spatial attention mask helps the manipulation to focus only on the target attribute specific area. A cycle reconstruction and the corresponding cycle consistency loss are also included in the training process.

Pumarola *et al.* [45] adopts a very similar method to utilize the attention mask to find the most important areas for editing in GANimation. The difference lies in that GANimation aims to edit facial expression, with the continuous facial action unit labels as the target expression code, enabling expression editing in a continuous manifold.

Different from the traditional cross-domain models, Choi *et al.* [13] implements a multi-domain image-to-image translation model starGAN by a single generator. Conditioned on the target domain label, starGAN can transfer the input image into the target domain, therefore can implement facial attribute editing of multiple attributes. To encourage the generator to translate the image into the correct domain, a domain classifier is implemented on top of the discriminator to classify which domain the real/fake

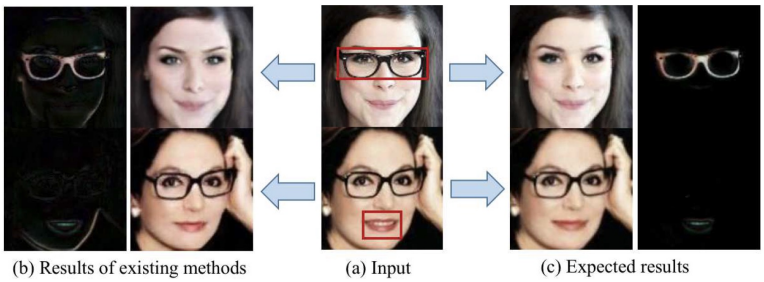


Figure 2.10: **Spatial attention mask**: helps the model to focus only on modifying attribute-specific area while keeping other regions unchanged. Figure taken from [66].

image belongs to. The training process is illustrated in Figure 2.11. The generator G takes in the input image and target domain label, to produce the fake image, which is later fed into G with the original domain label to reconstruct the input image, leading to a cycle reconstruction loss. The produced fake image and the true sample image are fed to the discriminator, to distinguish the authenticity and for the domain classification step.

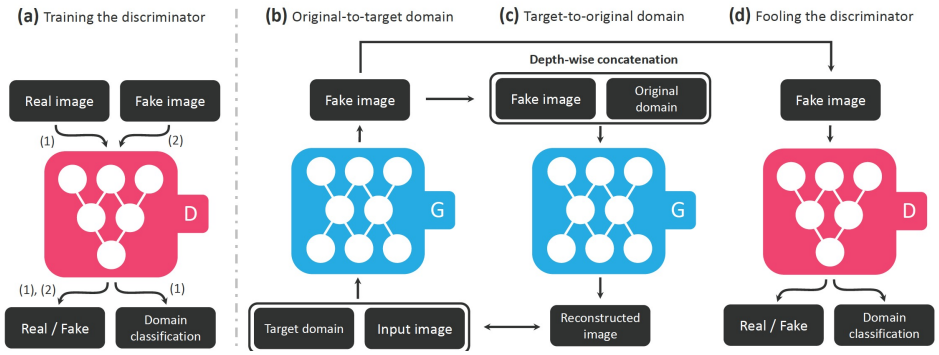


Figure 2.11: **Architecture of starGAN**: a cycle reconstruction is implemented for supervision, with an additional domain classifier on top of the discriminator to encourage the generator to map the input into the correct domain. Figure taken from [13].

Disentanglement-based Besides the cycle reconstruction method, there are many research works trying to manipulate face attributes by disentanglement of latent space [59, 19, 49]. The attributes are encoded in an disentangled manner in the latent space, and exchanging the attribute in the latent space will generate a new image with target attributes.

Xiao *et al.* [59] proposes the ELEGANT model, taking a positive image A with i-th attribute, a negative image B without i-th attribute as inputs, which are fed to the generator for encoding in the latent space. To formally describe, the latent encoding for A and B are denoted by z_A and z_B respectively:

$$\begin{aligned} z_A &= E(A) = [a_1, a_2, ..., a_i, ..., a_n] \\ z_B &= E(B) = [b_1, b_2, ..., b_i, ..., b_n] \end{aligned} \tag{2.17}$$

where n attributes are predefined in total, and E is the generator. Then the latent encodings for this i-th

attribute in A and B are exchanged:

$$\begin{aligned} z_C &= [a_1, a_2, ..., b_i, ..., a_n] \\ z_D &= [b_1, b_2, ..., a_i, ..., b_n] \end{aligned} \tag{2.18}$$

The exchanged latent encodings are forwarded into the decoder to generate four residual images:

$$\begin{aligned} &\mathrm{D}([z_A, z_A]) = R_A, \quad A' = A + R_A \quad \mathrm{D}([z_C, z_A]) = R_C, \quad C = A + R_C \\ &\mathrm{D}([z_B, z_B]) = R_B, \quad B' = B + R_B \quad \mathrm{D}([z_D, z_B]) = R_D, \quad D = B + R_D \end{aligned} \tag{2.19}$$

Therefore the generated A', B' are expected to be the same with the input A, B, while C, D should be the edited result of A, B without and with the attribute respectively. The adversarial loss, and the reconstruction between A, A' and B, B' are included in the loss function. Each training step trains with one attribute and all the attributes are trained iteratively.

A similar method is proposed in [19] to exchange latent disentangled representations for attributes. As illustrated in Figure 2.12, the input image x_1, x_2 are sampled independently and fed to encoder for latent encoding. The latent feature chunks of x_1 and x_2 are swapped (referred as the mixing step) and forwarded to the decoder to generate x_3. The same process is repeated on x_1 and x_3 (referred as the unmixing step), and the synthesized x_4 is expected to be the same as input x_1, which is similar as a cycle reconstruction. An additional classifier is included in training to decide for each chunk whether the composite image is generated using the feature from the first or second input image, in order to encourage each feature chunk to encode different attribute information.

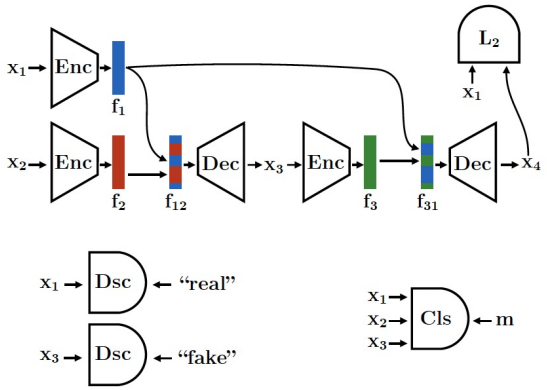


Figure 2.12: **Architecture of [19]**: the feature chunk of two input images are mixed and later unmixed for a cycle reconstruction. Figure taken from [19].

2.4 Pose Transfer

Pose transfer is an important topic in image synthesis, referring to image synthesis in unseen poses. Pose transfer can involve different image domains, and here we give a brief overview on human pose transfer and face pose transfer.

Human pose transfer Given a full body image of a person and a target pose encoded with keypoints of the human-body joints, the task aims to generate a new image of the same person but in the target pose [36, 37, 5, 30].

The first research work to approach human pose transfer is [36], which incorporates two stages of training. During the first training stage, a coarse generator G_1 takes in the input image and target pose to a U-Net structure to output a coarse output of the person in the target post. During the second training stage, the coarse output from G_1 is concatenated with the input image, and then fed into the second generator G_2 to produce a difference map, which is later added to the coarse output to generate the final result with fine details, with G_1 fixed during the second training stage. The pose is encoded by a 18-channel keypoint heatmap, representing 18 body keypoints.

Later Ma *et al.* [37] proposes another approach to implement human pose transfer task based on a two-stage reconstruction pipeline to learn a disentangled representation of foreground, background and pose, with body keypoints as the structure information. During the first stage, the input image x is fed into multi-branch encoders to encode information of background, foreground and pose in a disentangled space respectively, and then the disentangled factors are combined together for reconstruction of the input image. To disentangle the pose and foreground appearance, bounding boxes are used to extract region of interests from the foreground, and each small region is passed to the model separately. For the second stage, three mapping functions to map the Gaussian noise into the continuous feature embedding space are learned, and then a pretrained decoder is used to map the feature embedding into the image. Fake embeddings are generated from the noise for the adversarial training, and the discriminator aims to distinguish the fake embeddings from the real embeddings, as illustrated in Figure 2.13. Such disentanglement eliminates the requirement for paired data during training, but only need to reconstruct the input image. Such a sampling strategy enables manipulation over the foreground, background and pose, allowing more control over the output.

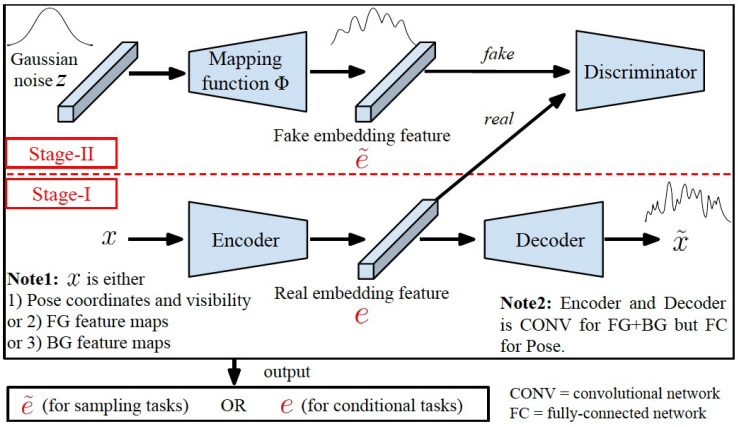


Figure 2.13: **Architecture of [37]**: the training includes two stages: the first stage aims to disentangle the input image into foreground, background and pose in the embedding space, and use the decoder for reconstruction, while the second stage learns mapping functions to map the noise into the fake embedding space, and then transformed into the output image by the pretrained decoder. Figure taken from [37].

Different from the idea to disentangle the image factors in the latent space, Balakrishnan *et al.* [5] directly utilizes a spatial transformation module to spatially move the body parts towards the target pose. The model includes four modules: Module A aims to perform image segmentation for the source image, to generate a segmentation mask for each body part and the background. Module B takes the body part masks as inputs, as well as the source pose and target pose, to spatially transform the body parts into the target pose. A separate geometric transformation is directly computed from the input poses (but not

learned during training), for warping the input body segmentations. Module C takes the transformed body segmentation parts to generate the foreground, while in Module D, the background is separately filled with appropriate texture. Finally the output of Module C and Module D are combined together to produce the new image in the target pose. All modules are U-Net shape-like, and trained jointly. However, such training strategy will require paired data for supervised learning.

Besides the calculation in the 2D plane, some research work proposes to estimate dense and intrinsic 3D appearance flow to better guide the transfer of pixels [30]. A 3D model is fitted for the source and target pose pair, and then projected back into the 2D plane to compute the dense appearance flow. For training, the model requires a source and target image pair of the same person as inputs, together with their corresponding pose annotations, which is encoded by a 18-channel binary heatmap, representing 18 human keypoints respectively. Let (x_1, x_2) denote the input source and target image, and (p_1, p_2) denote the input pose and target pose. A flow regression module is first trained to compute the appearance flow map F and the visibility map V to jointly represent the pixel-wise correspondence between x_1, x_2 in 3D space:

$$\begin{aligned} f_i &= F(u_i) = u_i' - u_i \\ v_i &= V(u_i) = visibility(h_i, x_1) \end{aligned} \tag{2.20}$$

where u_i', u_i are the 2D coordinates in x_1, x_2 which are projected from the same 3D point h_i, and $visibility(h_i, x_1)$ indicates whether the point h_i is invisible in x_1. After the flow regression module is trained, as illustrated in Figure 2.14, the source image x_1 is fed into the appearance decoder, and feature warping module, while the target pose p_2 goes into the pose encoder. The decoder takes the appearance and multi-level features to generate the a coarse output. The source image is combined with the coarse output for pixel warping for refinement to generate the final output image in the target pose. The objective function includes the adversarial loss, reconstruction loss of x_2 and a perceptual loss.

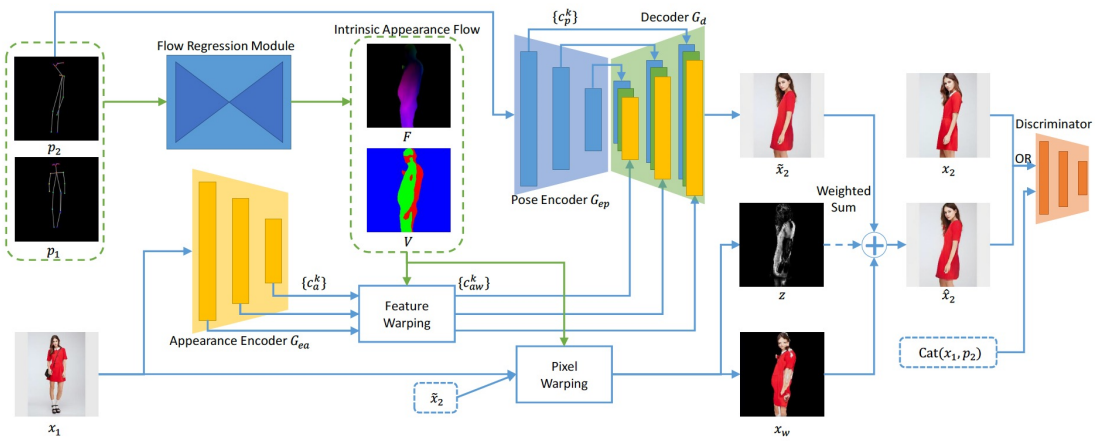


Figure 2.14: **Architecture of [30]**: a flow regression module is first trained to compute the appearance flow map F and the visibility map V, which can represent pixel-wise correspondence between the source image and target image. Figure taken from [30].

Face pose transfer Face pose transfer and face frontalization have gained much research interest as it can facilitate pose-invariant face recognition [52, 44, 65]. Most of the works are based on the disentanglement of head pose and appearance feature.

Peng *et al.* [44] utilizes a 3D facial model to generate a series of pose-variant faces from a near-frontal image. The 3D shape of the face is reconstructed by the 3DMM model from a near-frontal face, to embed the face image into low-dimensional parameter space which characterizes the shape features. Then multi-source supervision is used to learn a feature embedding, which is separated into the identity feature and non-identity feature. The non-identity feature is further branched into representations of head pose and facial landmarks. Paired images are required during training stage, to exchange the non-identity feature of the two input images in the latent space. Features after the exchange will be sent into the decoder for image reconstruction.

Tran *et al.* [52] trains an auto-encoder for image synthesis, with the encoder to produce image embedding which characterizes the appearance information, then combined with the pose code and noise, and fed into the decoder for synthesis with the given target head pose. In order to improve the pose consistency and identity consistency, the discriminator is trained not only to distinguish the generated faces and real faces, but also to predict the identity and pose of a given face.

A similar method is implemented in [65], but with additional expression code as the input for a joint modeling of pose and expression. A classifier is included in the training to predict the expression label. However, the defined expression here only includes several discrete labels, therefore cannot perform continuous expression transformation.

2.5 Face Reenactment

Here we roughly divide the existing methods for face reenactment into two categories: model-based methods [51, 25, 12, 50, 54, 14, 16] and feed-forward-based methods [25, 57, 56, 22, 60, 69]. The model-based methods typically have three steps: 1) Capture face movement by a face tracking or using optical flow on either RGB or RGB-D camera. 2) Fit the movement of the face into a parametric space or a 3D model. 3) Once the model is fitted, the final stage is to re-render a new video.

The feed-forward method has a similar pipeline but replaces the third re-render step with a neural network generator. For example, Kim *et al.* [25] takes 3D rendering results, illumination and the expression as inputs to generate a near-real quality image. Besides the 3D methods, the existing 2D face reenactment method involving the neural network generator can be divided into two lines: target-specific methods and one-shot methods.

Target-specific methods The target-specific methods refer to architectures which implement the subject-specific generation. In this setting, a large amount of training data is required for the target person, and a separate training process is required for each single target face.

The project Deepfakes [2] drew a lot of attention earlier for its realistic synthesis. Generally speaking, Deepfakes implement face reenactment by training a network to recover the target person's warped face to the original face by an auto-encoder, as shown in Figure 2.15. For target face A, the real face is first warped, then fed to the auto-encoder to reconstruct face A. Here the encoder is shared among all different faces while the decoder is target-specific, with specific information of A. During the test phase, face image of a random person B will be fed into the auto-encoder to transform B's face into A while keeping the expression unchanged. The learning objective includes the reconstruction loss of face A, the adversarial loss and a perceptual loss.

Instead of directly feeding the face encoding into the target-specific decoder to reenact the target face, which might result in structural artifacts, Wu *et al.* [57] takes the face boundary information as the input to achieve a competitive result in ReenactGAN. In addition to the shared encoder and target-specific decoder, a target-specific boundary transformer is trained to bridge the identity gap between the source face and target face. The source face with the target expression and pose is first mapped into a boundary

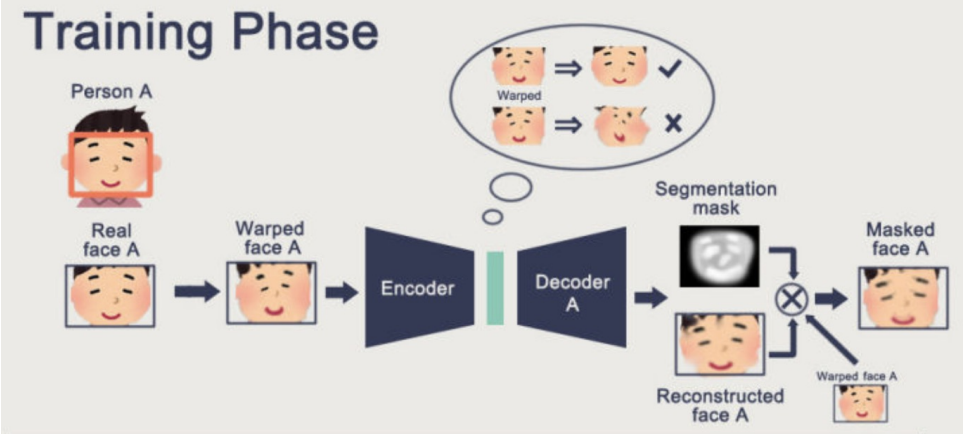


Figure 2.15: **Training phase of Deepfakes**: an auto-encoder is trained to reconstruct the original face from the warped face input. Figrue taken from [3].

latent space, and the boundary transformer is used to adapt the source face boundary to the target face boundary. Transforming the source face into the target face in the boundary latent space is easier to train than directly transforming in the pixel space, and can perform photo-realistic face reenactment.

Despite the satisfying synthesis results, target-specific face reenactment methods require much data and a separate training for each target face, therefore imposing high requirements for users' expertise and computational resources.

One-shot methods One-shot face reenactment refers to models which requires only one image input for each target face without the necessity to collect additional data nor extra training process.

The existing facial attributes manipulation methods [13, 11, 63, 10, 61, 45] support single image input. However, they only alter the appearance of facial components and face difficulty in handling large shape and pose changes. Meanwhile, there are one-shot models that generate images under the guidance of texture and semantic information, such as full-body generator [36, 37, 5, 30, 15, 36, 65, 52] and the street view generator [43]. These methods are confronted with identity shift or coarse details of face when adopted to generate face image directly.

There have been several trials in one-shot generative methods recently [6, 7, 67, 31, 64]. A disentangle learning framework is proposed in [7] to implement face synthesis with one image as appearance reference and another image as the pose guide. However, the problem setting here is a bit different from the face reenactment: only the identity information from the reference will be preserved in the output image, while all other attributes will be inherited from the pose guide image, including illumination and background. Two independent encoders are adopted to encode the identity and non-identity features respectively, with one generator to combine these two features for identity-preserving synthesis. A KL loss is included for the non-identity feature encoder to suppress the identity information in this branch, and an identity classifier participates in training to restrict the identity consistency between the generated result and input reference image.

Facial landmark is utilized in both FaceSwapNet [67] and [64] for pose guiding in face reenactment. FaceSwapNet designs a landmark swapper module to overcome the structural gap between the source face and the target face, similar as ReenactGAN [57]. Two landmark encoders first map the landmarks from the target face and source face into latent vectors, then a decoder maps the latent vectors into the transformed landmark. The resulting landmark combining the identity information from the target

face and expression from the source face are then fed into a landmark-guided generator for synthesis. However, FaceSwapNet can only reenact expressions but cannot transform haed pose as well.

On the contrary, Zakharov *et al.* [64] performs few-shot face reenactment which can handle both expression and pose transformation. The model architecture is illustrated in Figure 2.16. A few head images of the same person extracted from the same video are fed into the embedder to produce the embedding vectors for pose-independent information. The target pose landmark goes into the generator, which takes the embedding vectors as adaptive instance normalization parameters, to generate the output image in the target pose. The objective function includes the perceptual and adversarial losses, and a content loss which is the reconstruction loss between input pose guide image and the generated output. The architecture here can synthesize photo-realistic images, but still require paired data from videos for training, which can be usually hard to obtain. Besides, the pose guide in most of the test cases come from the target person, therefore the identity shift problem may exist.

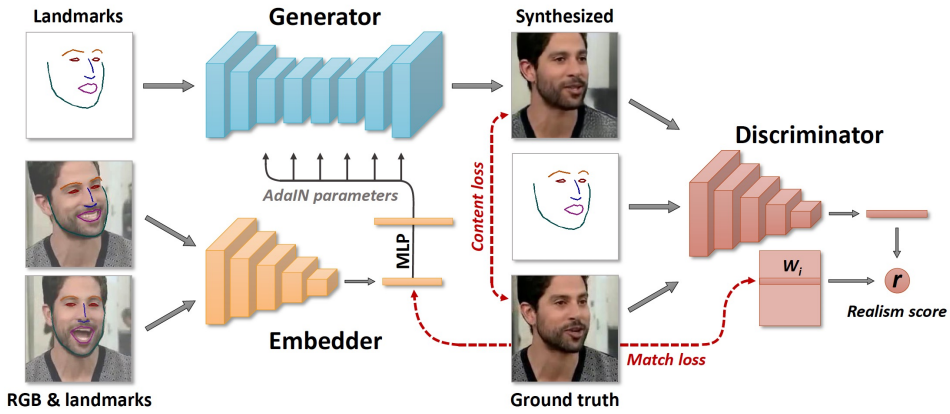


Figure 2.16: **Training framework of [64]:** paired-images are utilized for training with facial landmarks as the pose guide input. Figure taken from [64].

One-shot pose transformation in other domains are explored in [31] for pose reenactment in animals. A similar architecture is adopted here, to use two separate encoders to encode the reference image and pose guide image independently, and one unified decoder for synthesis. The training is implemented as a weak-supervision setting with unpaired data, but the performance in human face domain is not demonstrated.

One-shot face reenactment has been a difficult task due to the challenges in large shape changes, the complexity of pose and expression, and the need for realistic facial details and identity consistency, which motivates us to explore this topic in depth. In particular, we propose a one-shot face reenactment framework which only requires one reference image as the input for each target person during both training and testing stage. In addition, we train this framework in a weakly-supervised method. As far as we know, out model is the first work to approach the one-shot face reenactment problem with no need for paired images during training, nor separate training for each target people. The proposed model can achieve the consistency of head pose, expression and identity with one-shot input in one unified model.

Chapter 3

Methods and Architectures

3.1 Model Intuition

To clarity the task, the problem setting is explained as the following:

Given two images as the inputs, one called pose guide image I_{pose}, containing the source expression and pose, the other one called reference image I_{ref}, which are images of the target person with corresponding appearance and identity information. The task is to transfer the source expression and pose in I_{pose} to the target face I_{ref}. Therefore the output I_o should be the target person face in I_{ref} with the same expression and head pose as in I_{pose}, see Figure 3.1.

Images can be described by a set of attributes in the latent space, which are further divided into two parts in our task: shape information $b \in \mathcal{B}$ (expression and pose), and appearance information $a \in \mathcal{A}$ (all attributes except shape, including identity, makeup, etc.). A face reenactment model is expected to separate these two parts of attributes, to incorporate appearance information a from I_{ref} and b from I_{pose} to generate the final I_o.

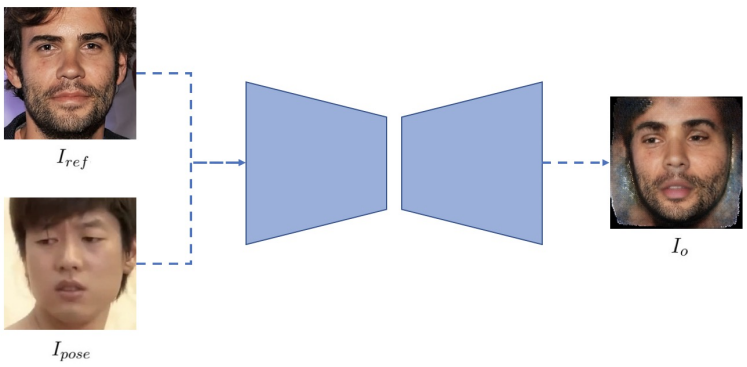


Figure 3.1: **Problem setting**: the task aims to transfer the expression and pose of pose guide image I_{pose} to the reference image I_{ref}, without changing any other attribute in I_{ref}.

With the goal to transfer the source expression and pose to the reference face image, the intuitive method is to utilize two separate encoders to encode the pose guide image and reference image into two disentangled latent space, with one joint decoder for reenactment. Specifically, for any input face $I \in \mathcal{I}$, we wish to first encode it into the Cartesian product of appearance representation $a \in \mathcal{A}$ and the shape representation $b \in \mathcal{B}$ through an encoder $F : \mathcal{I} \to \mathcal{A}$ and $E : \mathcal{I} \to \mathcal{B}$. Then we need a general decoder

$D : \mathcal{B} \times \mathcal{A} \rightarrow \mathcal{I}$ to map the appearance and shape from the latent embedding space back to the face space. For the input images,

$$\begin{aligned} I_{ref} &= D(a_{ref}, b_{ref}) = D(F(I_{ref}), E(I_{ref})) \\ I_{pose} &= D(a_{pose}, b_{pose}) = D(F(I_{pose}), E(I_{pose})) \end{aligned} \tag{3.1}$$

Therefore the reenacted face is generated by computing:

$$I_o = D(a_{ref}, b_{pose}) = D(F(I_{ref}), E(I_{pose})) \tag{3.2}$$

However, the difficulty lies with how to encode two input images properly, to disentangle the appearance information and shape information in latent space respectively. Unlike previous work, we propose to use face parsing maps as the latent representation to encode the shape information of the pose guide image I_{pose}, and an auto-encoder to encode appearance representation for better identity preserving performance. Figure 1.1 illustrates the three spaces $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{I}$ learned by our framework.

3.2 Disentangle-and-Compose Framework

Figure 3.2 shows the overview of the proposed one-shot reenactment model. We first disentangle the shape and appearance by a shape encoder (E) and an appearance auto-encoder (F), respectively. Then we progressively compose the source expression and the target identity into a realistic face. In particular, the decoder D is composed of multiple intermediate blocks (a yellow box and a green box in Figure 3.2 form a block). Each of the blocks concatenates feature maps from F that captures the appearance information with feature maps of the previous layer in D. Each block further uses the SPADE block [43], which takes face parsing maps as the input, to transfer the spatial information originated from the source face. To make full use of the encoding information, we insert intermediate blocks of different scales, taking multi-scale appearance encodings and face parsing maps as inputs.

3.2.1 Shape Encoder E

In the shape encoder E, the pose guide input I_{pose} is fed to a facial landmark detector to extract 106 facial landmarks, and additional 40 gaze landmarks to model the gaze movement of the face (see Figure 3.3).

With 146 facial landmarks, the landmarks of each facial component are transformed to a boundary map. Then we fill different colors in different facial parts (nose, eyes, etc.) of the boundary map to produce a 9-channel face parsing map, with each channel representing the segmentation mask of face contour, brows, eyes, gazes, nose, upper lip, lower lip, teeth, and nose bridge respectively. The pipeline of the shape encoder E is demonstrated in Figure 3.3.

The shape encoder is pretrained, and frozen during the training of the following two modules, which are Appearance Auto-Encoder (F), and Semantically Adaptive Decoder (D).

3.2.2 Appearance Auto-encoder F

Recall that appearance encoding is crucial for one-shot face reenactment. The appearance encoder needs to capture sufficient information, including identity information and local facial texture details, from just a single target image. Rather than concatenating the reference face to intermediate layers of D directly, we adopt the feature maps from an appearance auto-encoder, F, which takes the reference face I_{ref} as the input.

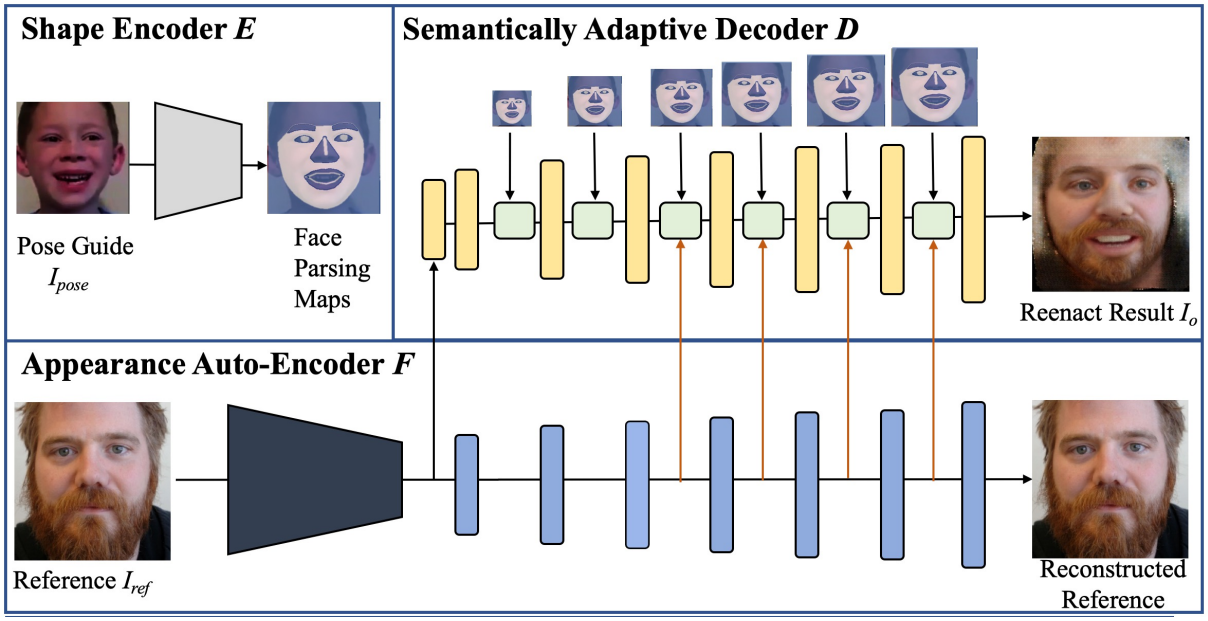


Figure 3.2: **An overview of the proposed one-shot reenactment model:** given a target image (the input appearance) and a source image (the pose guide) as input, we extract appearance features of the target image with F and feed the source image to E to generate face parsing maps simultaneously. The decoder D composes shape and appearance information to generate the reenactment result, which preserves the reference's appearance and the pose guide's shape. In order to improve the reenactment result, we insert SPADE residual blocks [43] to preserve spatial information and concatenate the feature maps from the face reconstruct decoder to D in a multi-scale manner. The D and F are trained simultaneously while the E is pretrained.

Although a pretrained encoder is adopted for the shape encoder, we found that a learned model can better preserve low-level textures and identity information in comparison to a fixed appearance encoder. The output feature map of the encoder in F is utilized as the appearance embedding, to be further used for reenactment decoder D. Besides the feature map of the encoder, the appearance auto-encoder includes a decoder (the blue blocks in Figure 3.2) that aims to reconstruct the target image from the latent feature maps. However, only the appearance encoder itself cannot preserve identity information well, causing the so-called 'identity shift' problem. To better preserve the identity during face reenactment, multi-layer features (in four different scales in total) are extracted from the decoder in F and concatenated with the corresponding feature maps in the semantically adaptive decoder, D.

There are 8 blocks in the appearance encoder to gradually dowmsample the input reference image I_{ref} from 256x256x3 resolution to 1x1x1024 resolution in the latent space, and 8 symmetric blocks in the appearance decoder to upsample the 1x1x1024 feature map back to 256x256x3 reconstructed reference image. See Figure 3.5 for the details inside the blocks.

3.2.3 Semantically Adaptive Decoder D

The semantically adaptive decoder takes appearance latent feature maps of I_{ref} and the face parsing map of I_{pose} to reenact the output face. As can be seen in Figure 3.2, besides the appearance feature map encoded by the encoder in the appearance auto-encoder F, multi-scale feature maps of 4 different

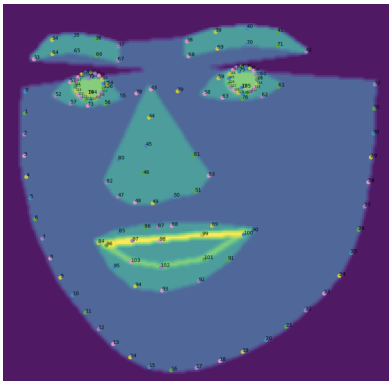


Figure 3.3: **Facial landmark annotation:** the landmark detector extracts 106 facial landmarks and additional 40 landmarks for gaze.

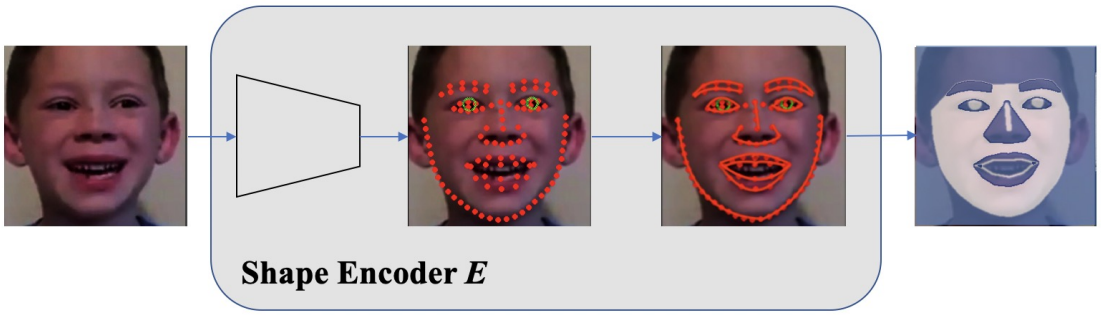


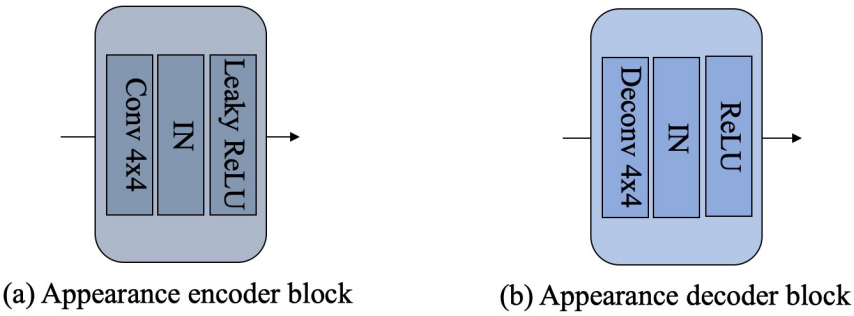
Figure 3.4: **Shape encoder E:** the shape encoder first extracts 146 facial landmarks, then transform them into boundary maps. Different colors are filled into the boundary maps to produce a 9-channel face parsing map.

spatial resolutions from the last four layers of the decoder in F are also concatenated into D, to recover facial details and identity feature from multiple resolutions. Also, the face parsing maps are fed into D in different spatial resolutions.

A U-Net structure is commonly used in the generation step [23, 55, 70] as its skip-connect structure can alleviate information loss. Normalization layers such as BatchNorm and InstanceNorm are also widely used to escalate model convergence.

However, as shown in [43], these commonly used normalization layers will uniform the spatial information in feature maps and cause semantic information loss. Inspired by [43], instead of stacking conventional blocks that include convolutional, or deconvolutional, and normalization layers to encode semantic information, we insert multi-scale SPADE blocks into the decoder D, to transmit spatial information from the face parsing map into the decoder directly by estimating normalize parameters from the parsing (see Figure 3.6). As such, the normalization parameters spatially correlate to the face parsing by convolutional layers, therefore eliminating the need for an encoder in a U-Net structure and leading to a network with a smaller size.

Each SPADE block has a residual structure, consisting of 3 blocks of [SPADE unit, ReLU, conv1x1], as indicated in Figure 3.7. The input feature map flows through 2 consecutive [SPADE unit, ReLU,



(a) Appearance encoder block (b) Appearance decoder block

Figure 3.5: **Appearance auto-encoder structure:** the appearance auto-encoder F consists of 8 convolutional blocks in the encoder and 8 deconvolutional blocks in the decoder symmetrically. Here IN denotes InstanceNorm.

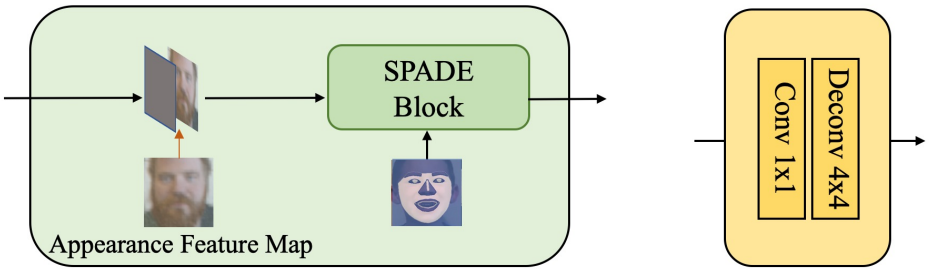


Figure 3.6: **Structures of the spatially adaptive decode D in details:** for each scale in D, feature maps from the previous layer are concatenated with corresponding appearance feature maps from the appearance auto-encoder F, and fed into the SPADE block, with the face parsing map fed into the SPADE block as well. Following the green block in each scale is a 1x1 convolutional layer and a 4x4 deconvolutional layer for upsampling.

conv1x1] blocks in the residual stream, with the bypass flowing through one additional [SPADE unit, ReLU, conv1x1] block. The final output is the sum of the residual stream output and the bypass output. Besides the feature maps, each SPADE unit takes the face parsing map of the corresponding scale as the second input.

The SPADE blocks are adopted to incorporate the input face parsing map into the reenactment decoder for face generation, and eliminate the effect that the BatchNorm or InstanceNorm layers tend to 'wash away' image-specific semantic information, therefore each SPADE unit takes the face parsing maps as input, and replace the learned scale and bias parameters in the InstanceNorm layers. Figure 3.8 illustrates the basic structure of the SPADE unit.

To be specific, let $m \in \mathbb{R}^{H\times\ W\times C}$ denotes the input feature map of the SPADE unit in the current layer, with width of W, height of H and C channels. Let $s \in \mathbb{R}^{H\times\ W}$ denotes the input face parsing map, with the same spatial resolution as m. Following the procedure of Spatially Adaptive Normalization idea in [43], the calculating is similar as InstanceNorm, to first obtain the mean and standard deviation of the

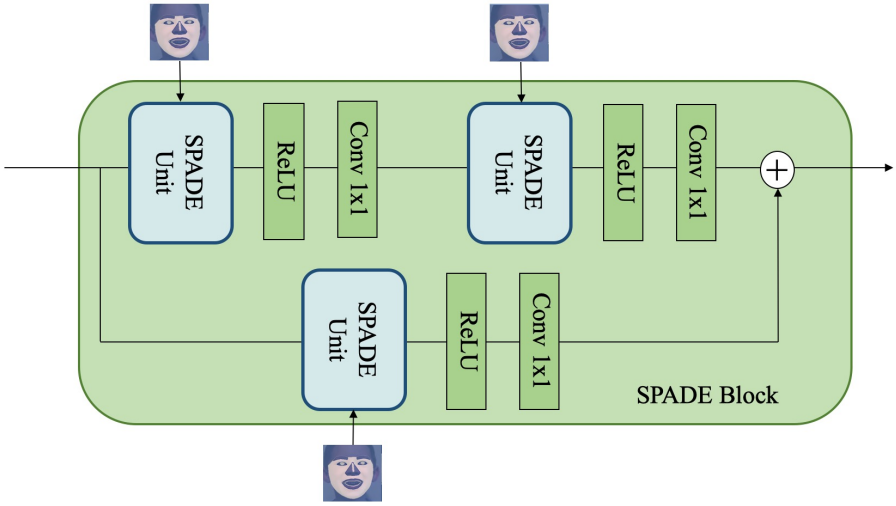


Figure 3.7: **Structure of the SPADE block:** for each SPADE block in D, the input feature map is fed into 2 [SPADE unit, ReLU, conv1x1] blocks, with another [SPADE unit, ReLU, conv1x1] block in the bypass to form a residual structure. Each SPADE unit also takes the face parsing map as input.

input feature map m in the channel-wise manner:

$$\mu_c = \frac{1}{HW}\sum_{i,j} m_{i,j,c} \tag{3.3}$$

$$\sigma_c = \sqrt{\frac{1}{HW}\sum_{i,j} m_{i,j,c}^2 - \mu_c^2} \tag{3.4}$$

where μ_c and σ_c denotes the mean and standard deviation of m in channel c. Usually, m will be normalized by μ_c and σ_c, and multiplied by a scale parameter, then added by a bias parameter, which are both learned during the training phase. Different from this procedure, we replace the learned scale and bias by two functions of the input parsing map s:

$$w_{i,j,c} = f(s) \tag{3.5}$$

$$b_{i,j,c} = g(s) \tag{3.6}$$

In implementation of our model, the mapping function f and g are two 3x3 convolutional layers applied to the input face parsing map. Then the output of the SAPDE unit is generated by:

$$w_{i,j,c}\frac{h_{i,j,c} - \mu_c}{\sigma_c} + b_{i,j,c} \tag{3.7}$$

With such architecture, there is not need to use the U-Net architecture for the generation branch, therefore eliminating the need to use an additional encoder in the U-Net, but only requires a decoder for reenactment generation, which is the semantically adaptive decoder D in our model. Stacking multiple SPADE blocks with face parsing maps and appearance feature maps of different scales as the input, D can better preserve facial appearance details, as well as utilizing the shape information from the pose guide image. D is trained jointly with the appearance auto-encoder F in an end-to-end manner.

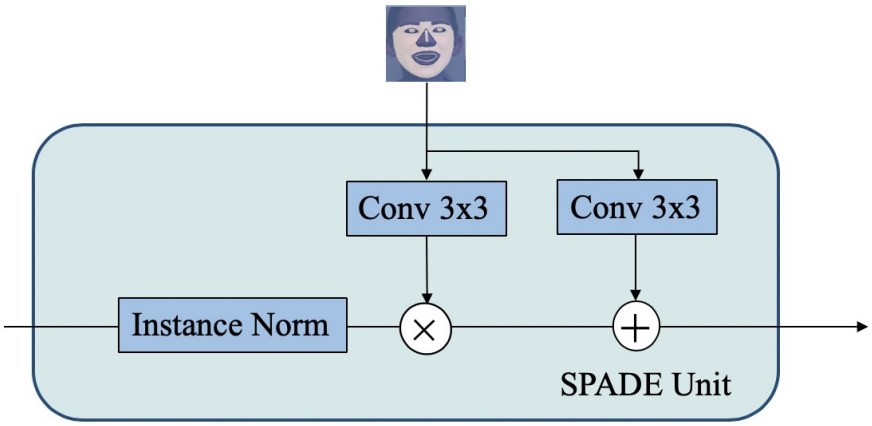


Figure 3.8: **Structure of the SPADE unit:** two learned parameters in the InstanceNorm layer are replaced by functions of the input face parsing map.

3.2.4 Boundary Resampling Strategy

Generally speaking, such face reenactment model would use paired-image as input to be trained in a fully-supervised way, with one image as the reference image I_{ref}, and the other image as the pose guide image I_{pose}. However, such paired images with different expressions or head poses, but of the same identity and taken under the same environment usually can only be collected from videos, and require much effort for pre-processing. In real-world applications, it is rather inconvenient to collect a large amount of such paired images if you need to train a model for a new face distribution. Considering such obstacles, we train this model in a weak-supervised way so that the model can be applied in most cases, as face datasets with only one image for each identity are available online, or can be easily collected. To be specific, during the training phase, the same image is used as both the reference image and the pose guide image.

However, note that the appearance encoder should stay independent from the shape information. To achieve this goal, we propose a simple but effective way to fill this gap. For each input image pair (I_{ref}, I_{pose}), instead of using the exact same image for both the appearance reference and pose guide, we replace the face contour of the input reference image with the face contour of another randomly sampled person during the training phase as the pose guide input, while keeping inner facial parts unchanged, referred as the 'boundary resampling strategy'. The reference input I_{ref} is warped into a new image $W(I_{ref})$ to fit the new face contour, therefore the pose guide image during training is represented by:

$$I_{pose} = W(I_{ref}) \tag{3.8}$$

In the implementation, the channel for face contour in the face parsing map of I_{ref} is replaced by the face contour channel of the randomly sampled person as the new pose guide, to encourage F to ignore shape information as much as possible, and force the independence between the appearance space $\mathcal{A}$ and the shape space $\mathcal{B}$. The face contour resampling is implemented by the traditional warping method, see Figure 3.9 for examples of the boundary resampling strategy.

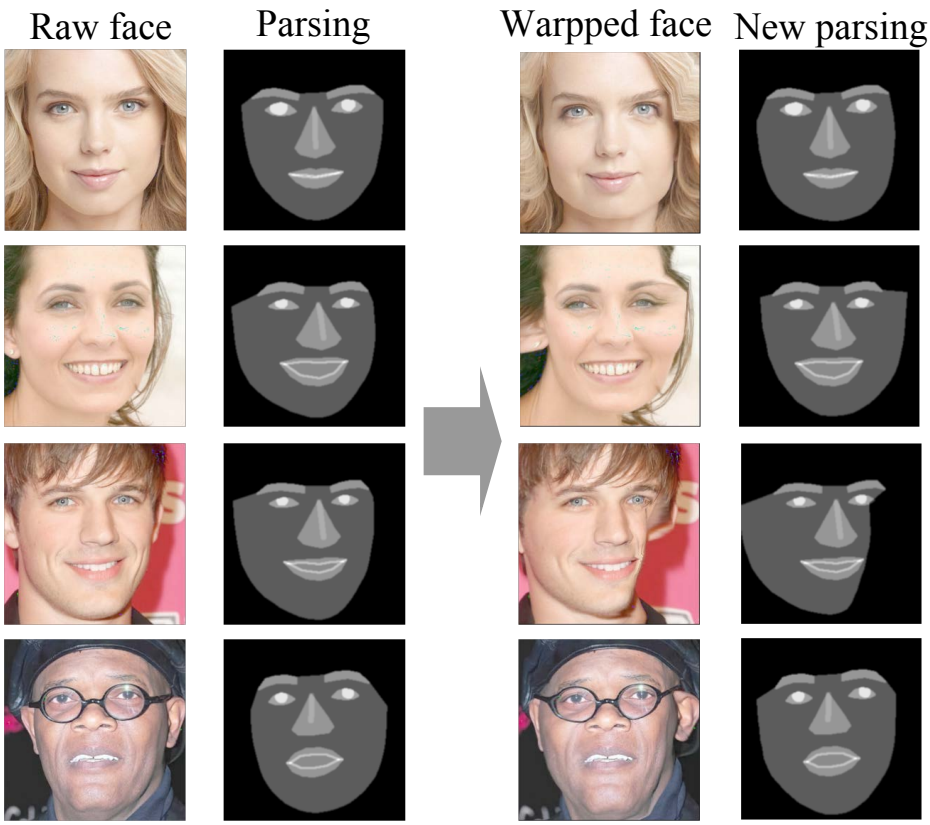


Figure 3.9: **Boundary Resample Strategy:** During training, the face contour of the reference image is replaced by the face contour of another randomly sampled person in the dataset, and the warped result is regarded as the pose guide image.

3.2.5 Loss Function

As mentioned earlier, instead of using paired data for training, we train the one-shot reenactment model in a weakly-supervised manner, using one image as both the reference image, and the pose guide image with the boundary resampling strategy. To better explain the loss function, Figure 3.10 illustrates the training setting, where the pose guide image is the warped reference image I_{ref} after boundary resampling, i.e., $I_{pose} = W(I_{ref})$.

The loss function includes two main parts: the loss from the appearance auto-encoder F, namely L_{app_recons}, and several losses from the semantically adaptive decoder D. Let I_o, I_{AE} denotes the output image, and the reconstructed image of F respectively. As the appearance auto-encoder F aims to reconstruct the input reference image I_{ref}, L_{app_recons} measures the difference between I_{ref} and reconstructed reference image I_{AE} by L1-norm loss:

$$L_{app\_recons} = ||I_{AE} - I_{ref}||_1 \tag{3.9}$$

The reenactment loss from the semantically adaptive decoder consists of four different losses, which are the reenactment reconstruction loss $L_{reconstruct}$, identity loss L_{id}, perceptual loss $L_{perceptual}$ and GAN generator loss L_{GAN}.

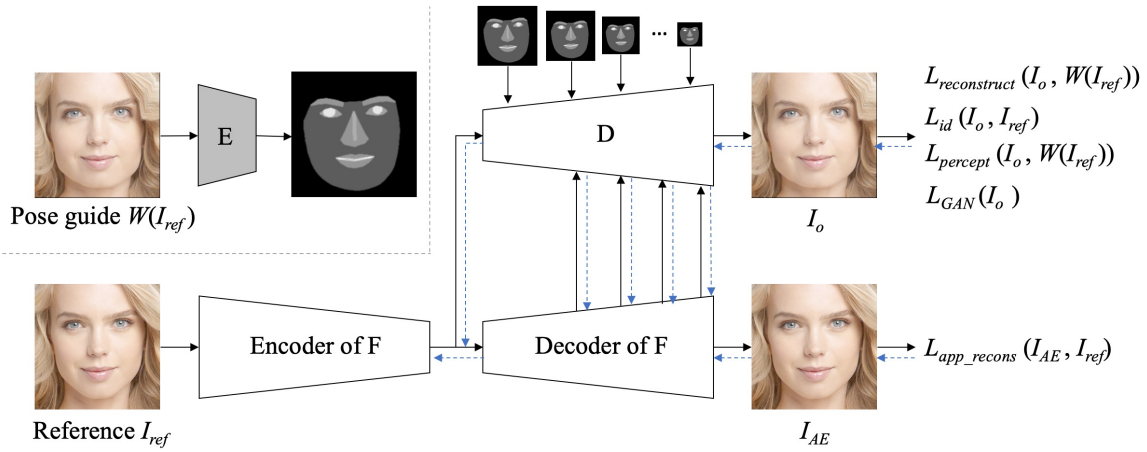


Figure 3.10: **Training setting:** during training stage, the pose guide image is the warped reference image after boundary resampling. The shape encoder E (the grey part) is pretrained and fixed during training stage, while other parts of the model is trained end-to-end. The black arrows denote the forward pass, while the blue dashed arrows denote the backward pass.

Reconstruction loss During the training stage, although the reference image is utilized as the pose guide by boundary resampling to encourage the disentanglement between the appearance and the shape space, the appearance information of the warped reference image $W(I_{ref})$ is assumed to be unchanged, i.e., the same as the appearance information from I_{ref}. Therefore the final reenactment result I_o should combine the shape information from $W(I_{ref})$, and the appearance information from $W(I_{ref})$ as well, leading to the $L_{reconstruct}$ measuring the reconstruction loss between I_o and $W(I_{ref})$:

$$L_{reconstruct} = ||I_o - W(I_{ref})||_1 \tag{3.10}$$

Identity loss The identity loss measures the identity information difference between the reenactment output I_o and input reference image I_{ref}, to force the reenactment result to preserve appearance and identity information from the reference image. A face verification model pretrained on CelebA [32] is adopted to extract the identity vector (the vector before the final prediction layer) of I_o and I_{ref} respectively. Let f represent the function to extract the identity vector of the pretrained face identification network, then the identity loss L_{id} is calculated by:

$$L_{id} = ||f(I_o) - f(I_{ref})||_1 \tag{3.11}$$

Perceptual loss The perceptual loss [55] measures the L1 loss between the reenactment result I_o and the pose guide of the relu1_2, relu2_2, relu3_3, relu4_3 features of the VGG-16 network [48]:

$$L_{perceptual} = \sum_{j=1}^{4} ||\phi_j(I_o) - \phi_j(W(I_{ref}))||_1 \tag{3.12}$$

where the function $\phi_j(j = 1, 2, 3, 4)$ denotes the feature extraction process in VGG-16 for relu1_2, relu2_2, relu3_3, relu4_3 respectively. The perceptual loss is widely used in image generation task for better reconstruction performance. See Figure 3.11 for details of the involved layers in VGG-16.

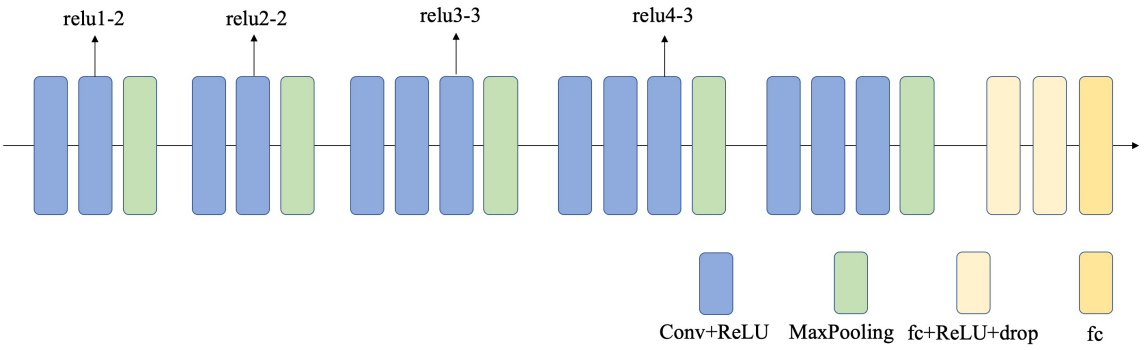


Figure 3.11: **Perceptual loss:** relu1_2, relu2_2, relu3_3, relu4_3 features of the VGG-16 network are extracted to compute the perceptual loss between I_o and $W(I_{ref})$, where fc denotes the fully connected layer.

GAN loss The LSGANs (least square GANs) loss is adopted in our implementation of the GAN loss. The least square loss is utilized in LSGANs to replace the cross entropy loss of the vanilla GAN, to push generated images which are far away from the decision boundary to the decision boundary, and fool the discriminator in the meantime. For the discriminator, a three-scale discriminator is used, which takes the spatial resolution of 64x64, 128x128, and 256x256 for each scale respectively (as illustrated in Figure 3.12). The discriminator is implemented by PatchGAN, which outputs a feature map instead of a single scalar, to achieve better authenticity local details and texture generation.

Let d_1, d_2, d_3 represent the three-scale discriminator of GAN, then the loss for the generator L_{GAN} and the discriminator L_d are:

$$L_{GAN} = \sum_{i=1}^{3} \frac{1}{2}(d_i(I_o) - c)^2 \tag{3.13}$$

$$L_d = \sum_{i=1}^{3} \frac{1}{2}[(d_i(W(I_{ref}) - a)^2 + (d_i(I_o) - b)^2] \tag{3.14}$$

where $a = 1, b = 0, c = 1$, and the three discriminators d_1, d_2, d_3 have equal weights in the loss function.

Overall loss With the definitions above, the overall combined loss function of our one-shot face reenactment architecture is defined as:

$$L_{all} = \lambda L_{app\_recons} + \alpha_r L_{reconstruct} + \alpha_i L_{id} + \alpha_p L_{perceptual} + \alpha_g L_{GAN} \tag{3.15}$$

where the balancing weights are set to $\lambda = 25$, $\alpha_r = 25$ and $\alpha_i, \alpha_p, \alpha_g = 1$. The semantically adaptive decoder D and the appearance auto-encoder F are trained simultaneously with the pretrained shape encoder E fixed, which means that D and F are optimized jointly while minimizing the overall loss L_{all}.

3.3 FusionNet

To eliminate disturbance from the background and to speed up convergence, the proposed face reenactment model focuses on inner facial components. Observing the results of our model, the inner facial

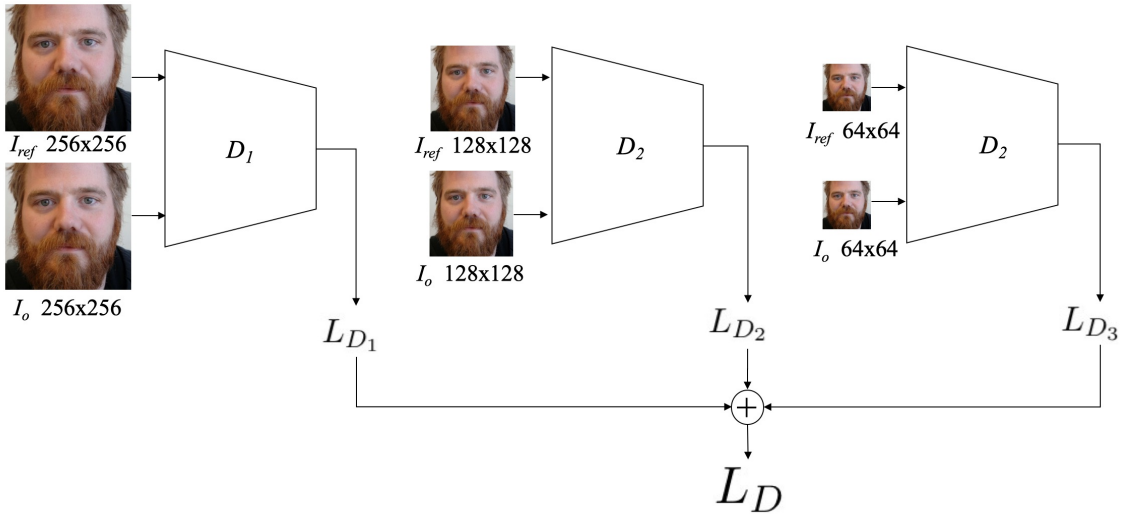


Figure 3.12: **3-scale Discriminator:** 3 discriminators which take images in resolution of 256x256, 128x128, 64x64 respectively are utilized in the adversarial training to improve the discriminative ability in multi levels.

components are inherently governed by the given pose guide face parsing. While this method is effective, the model still faces problems in generating facial texture details such as mustache and wrinkle. We note that warping-based methods [34, 62] has an advantage in generating facial details by warping pixels from the original image. Consequently, we propose the FusionNet, which takes the reenactment result of our model and the traditional warping result [1] as inputs, and generate a mask to fuse two generated images (Figure 3.13).

With the reference image I_{ref} and the pose guide image I_{pose} as inputs, the reenactment result I_o is generated by our proposed face reenactment model as described in Section 3.2. I_{warp} is generated by the traditional warping method, which can handle small pose changes. The pretrained shape encoder E is utilized to extract the face parsing maps for I_o and I_{warp} respectively, of which the product is the mask for fusion. The final fusion result is obtained by:

$$I_{fuse} = I_o \times (1 - mask) + I_{warp} \times mask \tag{3.16}$$

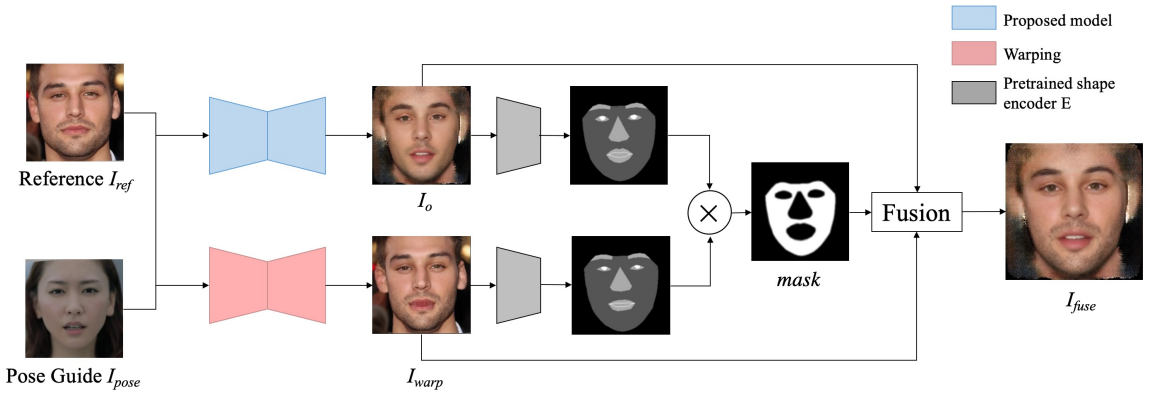


Figure 3.13: **FusionNet:** We propose FusionNet to leverage the face generated from multiple sources. Here we fuse the results from classical warping, which generates better facial textures, and works well on small pose changes; and our model, which is able to handle large pose changes and exaggerated facial actions.

Chapter 4

Experiments and Results

4.1 Experimental Setup

4.1.1 Training Dataset

For the training set, we use a subset of the CelebA-HQ [23] with 28k images. The high-quality of face images in CelebA-HQ helps the model to learn more details and generate realistic results.

CelebA-HQ CelebA-HQ [23] is a face dataset with 30k images, with high resolution of 1024x1024. It is a high-quality version of a subset of CelebA dataset [33]. A collection of JPEG images are firstly taken from the CelebA in-the-wild dataset, which are extremely varied in resolution and quality. Figure 4.1 illustrates the image processing pipeline described in the paper [23]. Each image is preprocessed by two pretrained networks, one auto-encoder to remove artifacts in JPEG image, and one 4x super-resolution network. Then the image is padded and Gaussian filtered to deal with cases where the face extends out of the image. After that, the image is aligned by selecting a rectangle area around the facial area with facial landmarks. Finally, the high-quality resampling is implemented to achieve the final 1024x1024 image.

(a) (b) (c) (d) (e) (f)

Figure 4.1: **Creating the celebA-HQ dataset :** the JPEG image (a) is fed into an auto-encoder to remove artifacts (b middle), and a 4x super-resolution network for higher quality (b bottom). Then the image is padded (c) and goes through a Gaussian filter (d). A rectangle area is selected in the image containing the facial region for alignment (e), and finally the 1024x1024 high resolution image is obtained by high-quality resampling. Figure taken from [23].

For the preprocessing in our experiment before training, the original image in CelebA-HQ with resolution 1024x1024 is first downsampled to 512x512, and then fed to the pretrained shape encoder E

to obtain the facial landmarks. The facial landmarks are utilized to align the 512x512 image, resulting in 384x384 aligned face images, and then cropped to 256x256 as the reference image inputs of the proposed model.

4.1.2 Testing Dataset

Considering the gaps between different datasets and in order to assess the general capacity of our one-shot reenactment model, we prepare three kinds of test datasets for the reference image input, each containing 500 images of different identities, with each photo aligned and resized to 256x256:

- **Same source:** Target face images from the same source, selected from CelebA-HQ with unseen identities during training.
- **Different source:** Target face images from a different source, selected from another high-quality face image dataset Flickr-Faces-HQ dataset (FFHQ) [24], with similar distributions of gender and ethnicity as Celeb-HQ.
- **In the wild:** Target face images in the wild, selected from a real-world dataset Real-world Affective Faces Database (RAF-DB) [29].

FFHQ is a high quality human face image dataset with resolution of 1024x1024. The images are crawled from Flickr, and automatically aligned and cropped using dlib. The dataset consists of 70k images, covering faces of different ethnicity and ages.

RAF-DB is a facial expression image dataset with around 30k human face images collected from the Internet, covering a large variation in ethnicity, expressions, head poses, etc.. See Figure 4.2 for samples data from FFHQ and RAF-DB.

Figure 4.2: **Sample images of the test dataset:** (a) FFHQ (b) RAF-DB. Figures taken from [24] and [29].

For pose guide data during testing, the faces are first detected by Faster R-CNN [47] from videos, then rigidly transformed into a mean shape and cropped to 256×256. 100 expressions that are unseen in the training set for each target face are sampled for the later quantitative evaluation, covering different expressions and head poses including yaw, pitch and roll degree. Figure 4.3 shows a subset of the sampled pose guide images.

Figure 4.3: **Sample images from the pose guide dataset:** the dataset used as pose guide images are detected from online videos, covering a large variety of expressions and head poses.

4.1.3 Implementation Details

The model is implemented using PyTorch, trained with batch size of 16. We use Adam [26] as the optimizer and set $\beta_1 = 0.5, \beta_2 = 0.999$. The training includes two stages: the main training stage and the finetune stage. During the main training stage, the model is trained with masked reference images with only facial region remained, and background removed for a better focus on inner facial components. The learning rates for the generator and discriminator are set to 1e-4 and 5e-5 respectively. The model is trained for around 45 epochs, which takes around 20 hours on 4 GPUs with 12GB memory. During the finetune stage, the model is finetuned with the full reference images for around 2 epochs to recover background information. The learning rates for the generator and discriminator are set to 2e-5 and 1e-5 respectively.

There are eight downsampling blocks in the appearance encoder, with each block containing 64, 128, 256, 512, 1024, 1024, 1024, 1024 channels. For the appearance decoder, there are eight upsampling blocks with each block containing 1024, 1024, 1024, 1024, 512, 256, 128 channels. The semantically adaptive decoder D includes eight upsampling modules, with the last four modules containing the SPADE blocks, and the first four modules only containing convolutional and upsampling layers.

4.1.4 Evaluation Metrics

For the quantitative comparison between the proposed method and other baseline methods, three evaluation metrics from different aspects are evaluated: (1) Facial AU Consistency. Expression consistency of inner facial components measured by facial action units [40]. 2) Head Pose Consistency. The consistency of head pose between the pose guide and the generated result. 3) Identity Preserving. The capability of preserving identity.

Facial AU Consistency. Facial action unit (AU) consistency is widely applied in studies on face generation [45, 57]. According to the Facial Action Unit Coding System, the facial muscles can be divided into different regions, and some facial regions will be activated and warped to a certain degree for each specific expression. The DISFA dataset [40] collects the facial action unit labels for different expressions. Instead of discrete labels, the action unit labels characterizes expressions in different degrees and different facial regions through continuous values formalized in vectors. A vector y_r to characterize a certain expression can be represented as:

$$y_r = (y_1, y_2, ..., y_N)^T \tag{4.1}$$

where N denotes the number facial regions the face is divided into, and each $y_i (i = 1, 2, ..., N)$ denotes the activation magnitude of the corresponding facial region, which is normalized to range $(0, 1)$. Therefore a single action unit or combinations of different action units will characterize a specific facial expression.

Here we evaluate the consistency between the generated images and the pose guide inputs following the protocol from [57]. 12 action units defined in DISFA dataset [40] are adopted for the evaluation. The 12 action units represent Inner Brow Raiser (AU1), Outer Brow Raiser (AU2), Brow Lowerer (AU4), Upper Lid Raiser (AU5), Cheek Raiser (AU6), Nose Wrinkler (AU9), Lip Corner Puller (AU12), Lip Corner Depressor (AU15), Chin Raiser and Mentalis (AU17), Lip Stretcher (AU20), Lip Part (AU25) and Jaw Drop (AU26) respectively. Diffrent combinations of action units can represent different expressions, as shown in Figure 4.4.

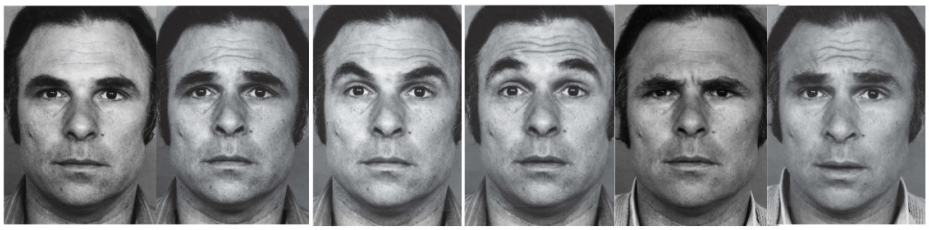


Figure 4.4: **Action unit based facial expressions:** based on a single action unit or different combinations of action units, different facial expressions can be characterized in a continuous manifold. Figure taken from [40].

For each pair of the input pose guide and the generated output image, the 12 action unit representations are first predicted by the pretrained AU prediction model, then the correlation coefficient is calculated upon the action unit representation between the two images, and a final mean correlation coefficient over all action units is obtained for the current test sample. An overall mean score of the correlation coefficient is reported as the facial AU consistency score. For this metric, the higher the better.

Head Pose Consistency. Besides the consistency of AU, we also evaluate the consistency of the head motion by head pose. The head pose characterizes the direction of the face, defined by three angles: row, pitch, yaw, as indicated in Figure 4.5. Yaw is the angle the face rotates around y-axis, pitch is the angle the face rotates around x-axis, and roll is the angle the face rotates around z-axis.

A pretrained head pose estimator model is applied to estimate the row, pitch, yaw angles (between 0 and 90 degrees) for each face image, in order to measure the consistency of row, pitch, yaw angles

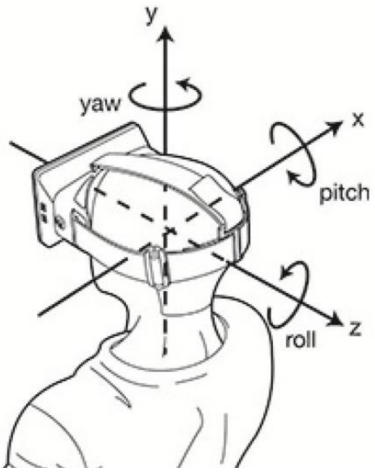


Figure 4.5: **Head pose:** defined by three angles, namely row, pitch, yaw. Figure taken from [28].

between the generated result and the input pose guide image. We report the average of mean absolute error of yaw, pitch, roll in degree, therefore for this metric, the lower the better.

Identity Preserving. Instead of using the perception loss, the identity preserving capability is evaluated by measuring the recognition accuracy with a fixed face verification model, which consists of a ResNet18 backbone and an MLP prediction layer. The corresponding identity features of the reference image and the generated result are extracted respectively, and a cosine similarity score is computed between these two identity feature vectors. See Figure 4.6 for the computing details. The generated face is judged to have the same identity with the reference input face if the cosine similarity score is above a threshold, then an overall accuracy is obtained. A higher accuracy score indicates a better identity preserving ability.

4.2 Single-image Method Baselines

Since there are few single-input face reenactment models which can be trained in the weakly-supervision manner, we implement several single-input generation models in other domains (human pose, cityscapes, landscapes, etc.) as our single-image method baselines, and train these models with our training data.

4.2.1 GANimation

Pumarola *et al.* [45] proposes GANimation as a conditional GAN based method which generates faces conditioned on target facial action unit annotations. Unlike earlier works in facial expression editing, which usually categorize different expressions into discrete and low number of classes, GANimation allows control over the continuous magnitude of activation of each action unit, and combination of several different action units. The model can be trained in an unsupervised manner to only utilize images with corresponding activated action unit annotations, with attention mechanism to change backgrounds or lighting conditions.

The problem formulation is defined as following: in an input image I_{y_r}, the facial expression is encoded by N action units $y_r = (y_1, y_2, ..., y_N)$, where each y_n denotes a value in $[0, 1]$ to characterize the activation magnitude of n-th action unit. Given a target action unit expression y_g, the goal is to transfer the input face I_{y_r} to image I_{y_g}, which has the expression the same as y_g characterizes, but

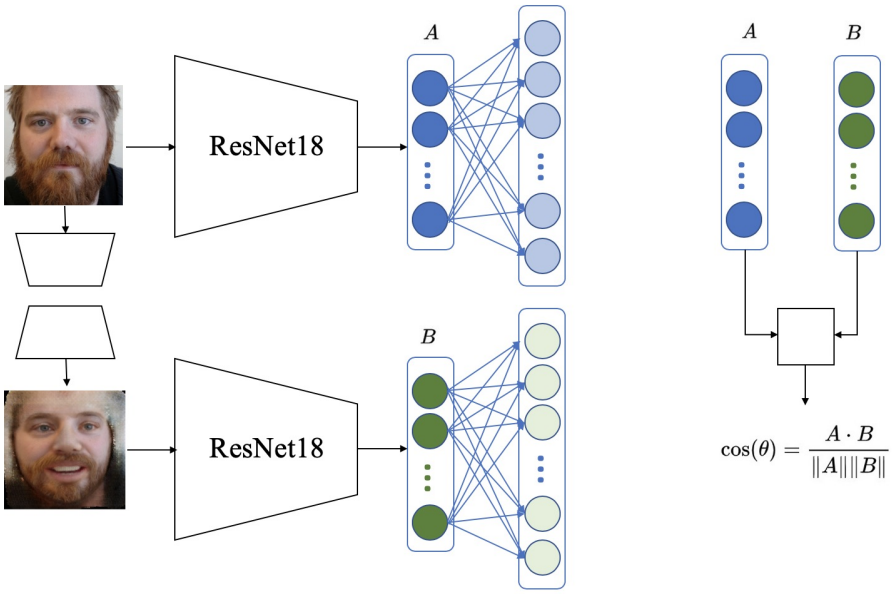


Figure 4.6: **Identity preserving evaluation:** the input reference image and the generated face are fed to the pretrained face verification model respectively, to extract corresponding identity feature vector before the final MLP layer. A cosine similarity score is computed between the two identity feature vectors, to measure the identity similarity between the input reference face and the generated face.

without changing the identity or appearance information, i.e., to learn a mapping f:

$$I_{y_g} = f(I_{y_r}, y_g) \tag{4.2}$$

Architecture The architecture of the generator in GANimation is shown in Figure 4.7. The input image I_{y_r} is concatenated with the action unit expression of the target expression y_g, and fed into the generator $G(I_{y_r}|y_g)$, for the edited image generation. The key idea of the generator design is to focus on facial regions involved in novel expressions, and keep the rest facial regions unchanged. Therefore an attention mechanism is added to generate an attention mask A in one branch, with the other branch generating a color mask C. The mask A indicates to what extend each pixel in C contributes to the final output, so that the generator can focus on pixels which contributes explicitly to the expression editing area. The output image I_{y_g} is fused from the two mask A, C, and input image I_{y_r}:

$$I_{y_g} = A \cdot C + (1 - A) \cdot I_{y_r} \tag{4.3}$$

$$A = G_A(I_{y_r}|y_g) \tag{4.4}$$

$$C = G_C(I_{y_r}|y_g) \tag{4.5}$$

where mask A is a 1-channel map with pixel values in range between 0 and 1, and mask C is a three-channel color map. The generator includes a cycle reconstruction, i.e., to feed the generated image I_{y_g} and the action unit annotation of the input image y_r, to reconstruct the input image I_{y_r}, to enable the model to be trained in an unsupervised way, with no requirement for image pairs of the same person under different expressions.

Besides the attention-based generator, GANimation utilizes two discriminators, namely D_I and D_y, with D_I to evaluate the authenticity of the generated image I_{y_g} by using PatchGAN [21], and D_y to

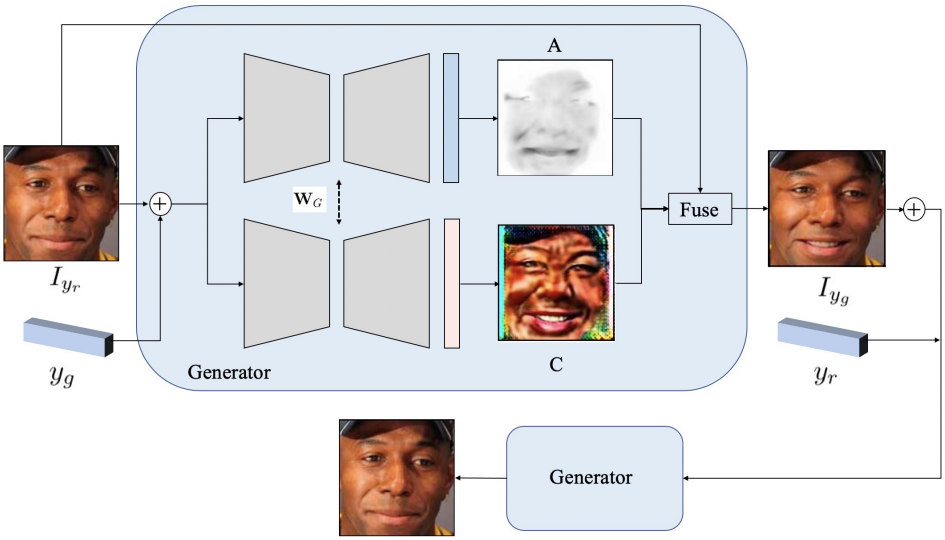


Figure 4.7: **Generator of GANimation [45]:** given the input image and the action unit annotation of the target expression, two generators are utilized to generate two masks, the attention mask A and the color mask C, to encourage the generator to focus only on facial areas involved in the expression changes, while keeping the rest facial regions unchanged. The final result is fused from A, C and the input.

predict $\hat{y_g}$, the action unit of I_{y_g} to encourage the expression consistency between the output image and input target action unit (see Figure 4.8).

Learning The model is trained in an unsupervised way in a cycle-reconstruction manner. The loss function includes four parts: the adversarial loss, the attention loss, the conditional expression loss, and the identity loss.

For the adversarial loss, the WGAN-GP [18] is utilized to replace the JS divergence in the original GAN [17] by the continuous Earch Mover Distance. Let $\mathbb{P}_r$ denote the input data distribution, and $\mathbb{P}_{\tilde{I}}$ denote the random interpolation distribution, then the adversarial loss is:

$$\begin{aligned} L_{adv}(G, D_I, I_{y_r}, y_g) = \mathbb{E}_{I_{y_r} \sim \mathbb{P}_r} \left[D_I \left(G \left(I_{y_r} | y_g \right) \right) \right] - \mathbb{E}_{I_{y_r} \sim \mathbb{P}_r} \left[D_I \left(I_{y_r} \right) \right] \\ + \lambda_g \mathbb{E}_{\widetilde{I} \sim \mathbb{P}_{\tilde{I}}} \left[\left(\left\| \nabla_{\widetilde{I}} D_{\mathrm{I}}(\widetilde{I}) \right\|_2 - 1 \right)^2 \right] \end{aligned} \tag{4.6}$$

During the training stage, the attention mask can easily saturate to 1, as it is learned from the gradients from other loss terms since there are no groundtruth for the attention mask, and the same situation also applies for the color mask, in which case the generator does not work at all. Therefore a total variation regularization term is used to force smooth spatial transformation in color to combine the two learned masks, A and C, and the input image for generation. An additional L2 norm loss is also included to penalize the weights of the attention mask A:

$$L_{att}(G, I_{y_o}, y_g) = \left[\sum_{i,j}^{H,W} \left[(A_{i+1,j} - A_{i,j})^2 + (A_{i,j+1} - A_{i,j})^2 \right] \right] + \|A\|_2 \tag{4.7}$$

Besides the discriminator D_I for the adversarial training, the other discriminator D_y predicts the action unit of the generated output image. To force the expression consistency between the output image

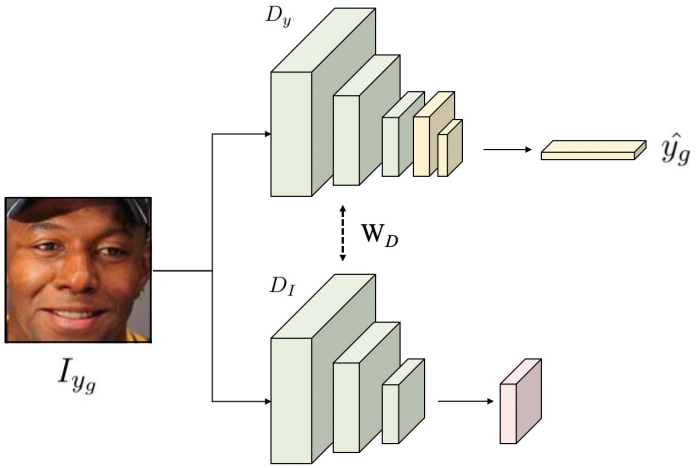


Figure 4.8: **Discriminator of GANimation [45]:** utilizes two discriminators, D_I and D_y, to evaluate the photo-realism of the generated results, and the expression consistency with the target expression annotation.

and target expression, a conditional expression loss is defined to calculate the error between the predicted action unit of I_g and the target expression annotation y_g, and the error between the predicted action unit of I_r and the expression annotation of the input image, y_r:

$$L_{exp}(G, D_y, I_{y_r}, y_r, y_g) = \|D_y(G(I_r|y_g)) - y_g\|_2^2 + \|D_y(I_{y_r}) - y_r\|_2^2 \tag{4.8}$$

A cycle reconstruction is built, by feeding the generated output I_{y_g}, and the action unit annotation of the input image, y_r into the generator to reconstruct the input image I_{y_r}. A cycle consistency loss is included in the loss function to guarantee the identity and appearance consistency between the input image and the output image:

$$L_{id}(G, I_{y_r}, y_r, y_g) = \|G(G(I_{y_r}|y_g)|y_r) - I_{y_r}\|_1 \tag{4.9}$$

Therefore the full learning objective can be defined as:

$$L = L_{adv}(G, D_I, I_{y_r}, y_g) + \lambda_1 L_{att}(G, I_{y_o}, y_g) + \lambda_2 L_{exp}(G, D_y, I_{y_r}, y_r, y_g) + \lambda_3 L_{id}(G, I_{y_r}, y_r, y_g) \tag{4.10}$$

4.2.2 PG2

Ma *et al.* [36] proposes the Pose Guided Person Generation Network (PG2) to synthesize person images in arbitrary poses, with a person image and the target pose as inputs, which is similar with our reenactment problem setting, but in a different domain. PG2 consists of two training stages: the first stage for pose integration, and the second stage for image refinement.

Stage 1 In the first stage, the model takes an image of the person (I_A) and the target pose (P_B) as inputs, and generates a coarse result with the person in the target pose, where the pose information is represented by 18 human pose keypoints extracted by a state-of-the-art pose estimator [8]. In our problem setting, the input I_A is the reference image, and the target pose P_B is replaced by the face parsing map

of the pose guide image. The generator in the first stage (G1) is a U-Net-like architecture with 6 blocks in the encoder and 6 symmetric blocks in the decoder (Figure 4.9), and a fully connected layer between the encoder and decoder. Skip connections are built between each pair of corresponding blocks in the encoder and decoder. Each block has a residual structure as illustrated in Figure 4.10.

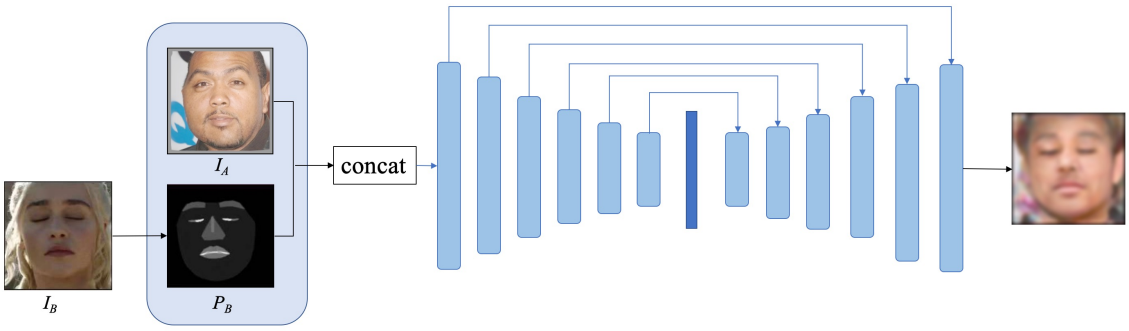


Figure 4.9: **G1 at stage 1 in PG2:** a U-Net-like architecture with 6 residual blocks in the encoder and 6 symmetric blocks in the decoder. The input is the concatenation of the reference image and the face parsing map of the target pose guide image. The middle layer in G1 is a fully-connected layer.

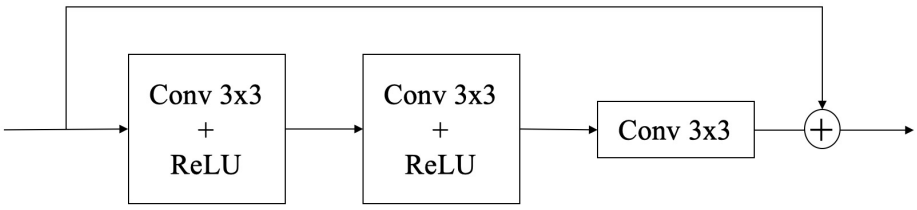


Figure 4.10: **Residual block for G1 in PG2**

PG2 is originally trained with paired-image for person image generation. Due to the lack of paired images taken from the same person under the same environment but with different expressions, we train the PG2 baseline for face reenactment with a pair of images take from the same identity. During the first stage, the loss function is:

$$L_{G1} = ||(G1(I_A, P_B) - I_B)(1 + M_B)||_1 \tag{4.11}$$

where M_B is a mask to remove the background information in I_B but only focus on inner facial components.

Stage 2 Stage 1 integrates the input reference image and pose guide to a coarse output, and captures low-frequency and structural information. The coarse output is fed to a second generator (referred as $G2$) in the training stage 2 for refinement. The generator (denoted by G2) in the second stage also has a U-Net-like structure, with four residual blocks (with the structure in 4.10) in the encoder and four symmetric blocks in the decoder respectively (see Figure 4.11). G2 takes the reference image input I_A and the coarse output from G1 (denoted by I_{B1}), and outputs a difference map to speed up convergence of the model. The final result is the sum of the difference map and I_{B1}. The loss for G2 is defined as:

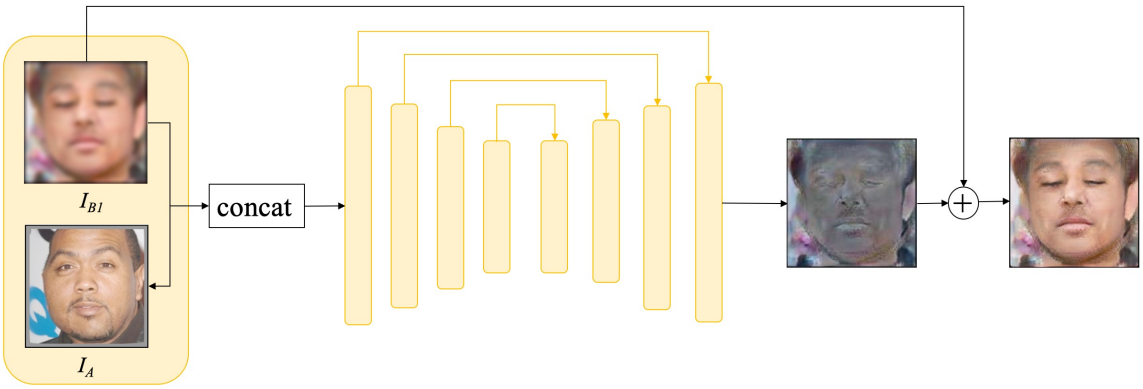


Figure 4.11: **G2 at stage 2 in PG2:** a U-Net-like architecture with 4 residual blocks in the encoder and 4 symmetric blocks in the decoder. The input is the concatenation of the reference image and the coarse output of G1.

$$L_{G2} = \lambda||(G2(I_A, I_{B1}) - I_B)(1 + M_B)||_1 + L_{GAN} \tag{4.12}$$

where L_{GAN} is the generator loss of GAN.

G1 is firstly trained with L_{G1} with batch size of 6, and learning rate of 2e-5 for 15000 steps. Then G2 is trained with L_{G2} and discriminator loss in GAN with $\lambda = 50$, batch size of 6, and learning rate of 2e-5 for around 40 epochs, with G1 fixed.

4.2.3 gauGAN

Part *et al.* [43] proposes gauGAN as an image style transfer method which allows user control over both semantic and style for synthesizing images with the target semantic map and style. Different from previous methods to directly feed the semantic map into the convolutional layers, gauGAN proposes to use the semantic map to represent the input layout, to modulate the activations in the normalization layers by spatially-adaptive transformation, which is also utilized in our proposed face reenactment model. The input semantic map replaces the two learned parameters in the InstanceNorm layer by two convolutional layers respectively, as explained in Section 3.2.3.

Architecture The model consists of a content encoder and a generator, as illustrated in 4.12. The encoder includes 7 [conv, InstanceNorm, LeakyReLU] blocks, and output the mean and variance of the distribution as shown in Figure 4.13 (a). The generator takes the predicted mean and variance, and the pose guide face parsing map, undergoes a series of SPADE blocks and upsampling layers to produce the reenactment result (see Figure 4.13) (b), where the SPADE block has the almost the same structure as in Figure 3.7.

Learning The baseline gauGAN model is trained in the same weak-supervision manner as the proposed reenactment model, with the perceptual loss, the reconstruction loss the same as our proposed model, the 3-scale GAN loss the same as [55], the additional KL divergence loss and feature matching loss from three scales of each discriminator. The KL divergence loss is expressed as:

$$L_{KL} = D_{KL}(q(z|x)||p(z)) \tag{4.13}$$

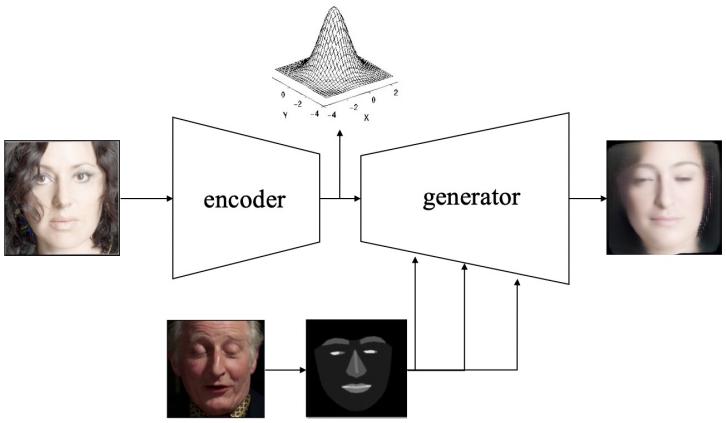


Figure 4.12: **Architecture overview of gauGAN:** mean and variance are learned by the encoder, which are fed into the decoder with segmentation masks for synthesis.

where $p(z)$ is the prior distribution and implemented as a Gaussian distribution. The variational distribution $q(z|x)$ is characterized by the mean and variance.

4.2.4 VU-Net

Esser *et al.* [15] proposes the VU-Net, which is a conditional generative model for shape-guided image generation by VAE. Instead of generating object images directly, VU-Net models the the intrinsic interplay of the shape and appearance. Given an input image x, the shape y and appearance z can be regarded as two latent variables to characterize x. Then a maximum a posterior estimate to maximize $p(x|y,z)$ can be viewed as an image generator conditioned on the shape and appearance information. Intuitively, VAE can be utilized to learn the generator $p(x|y,z)$. An approximate posterior $q(y,z|x)$ is introduced to obtain the evidence lower bound (ELBO), and to maximize the lower bound of the log-likelihood:

$$\begin{aligned}\log p(x) &= \log \int p(x,y,z)\mathrm{d}z\mathrm{d}y \\ &= \log \int \frac{p(x,y,z)}{q(y,z|x)} q(y,z|x) \\ &\geq \mathbb{E}_q \log \frac{p(x|y,z)p(y,z)}{q(y,z|x)}\end{aligned} \tag{4.14}$$

However, the joint prior distribution $p(y,z)$ is required in VAE, which is assumed to be a standard Gaussian distribution, while the goal is to disentangle the shape y and z in the latent space. Therefore we assume that a mapping exists between y and x: $\hat{y} = f(x)$. For example, f could be an edge detector or landmark detector. Given the mapping to estimate shape information, the task becomes to maximize a conditional likelihood $p(x|\hat{(y)})$:

$$\begin{aligned}\log p(x|\hat{y}) = \log \int_z p(x,z|\hat{y})\mathrm{d}z &\geq \mathbb{E}_q \log \frac{p(x,z|\hat{y})}{q(z|x,\hat{y})} \\ &= \mathbb{E}_q \log \frac{p(x|\hat{y},z)p(z|\hat{y})}{q(z|x,\hat{y})}\end{aligned} \tag{4.15}$$

Compared with the vanilla VAE objective in Equation 4.14, Equation 4.15 relies upon the conditional prior $p(z|\hat{y})$, which can be estimated from the training data. Instead of $p(y,z)$, the conditional prior is

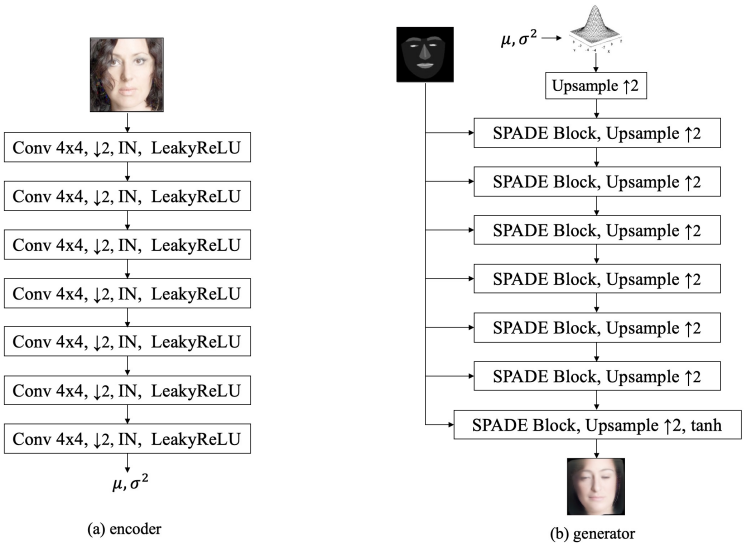


Figure 4.13: **Detail structures of gauGAN:** the IN here denotes the InstanceNorm layers.

more likely to characterize some interrelations between the shape and appearance information. Two networks, namely G_θ and F_ϕ, are implemented to estimate the two distributions, $p(x|\hat{y}, z)$ and $q(z|x, \hat{y})$, in Equation 4.15.

Architecture The model architecture is illustrated in Figure 4.14. The input image x and shape estimator $\hat{y}$ is fed into F_ϕ to model the appearance latent space distribution, and a sampled z from the estimated appearance space is concatenated with the shape latent embedding produced by the encoder in G_θ, and passed to the decoder in G_θ for image generation. G_θ is a U-Net-like architecture with skip connections between corresponding layers in the encoder and decoder in G_θ.

Learning Similar with our setting, the VU-Net can be trained in the self-supervision way, to reconstruct the input image x. The training loss includes a L1-norm reconstruction loss, with a KL divergence loss from VAE:

$$\mathcal{L}(x, \theta, \phi) = -KL\left(q_\phi(z|x, \hat{y})||p_\theta(z|\hat{y})\right) + \|x - G_\theta(\hat{y}, z)\|_1 \tag{4.16}$$

The VU-Net is originally proposed to synthesize object images, such as shoes or bags (conditioned on corresponding sketch images), human images (conditioned on human body pose landmarks). In the task of face reenactment, the same as our setting, we use face parsing maps to represent the shape estimator $f(x)$.

4.3 Comparison with Single-image Methods

We compare our proposed framework with all the four baselines mentioned above, and the result without FusionNet both qualitatively and quantitatively.

Figure 4.15 shows the result of all baselines and the proposed framework.

The characteristics of each baseline are described as following:

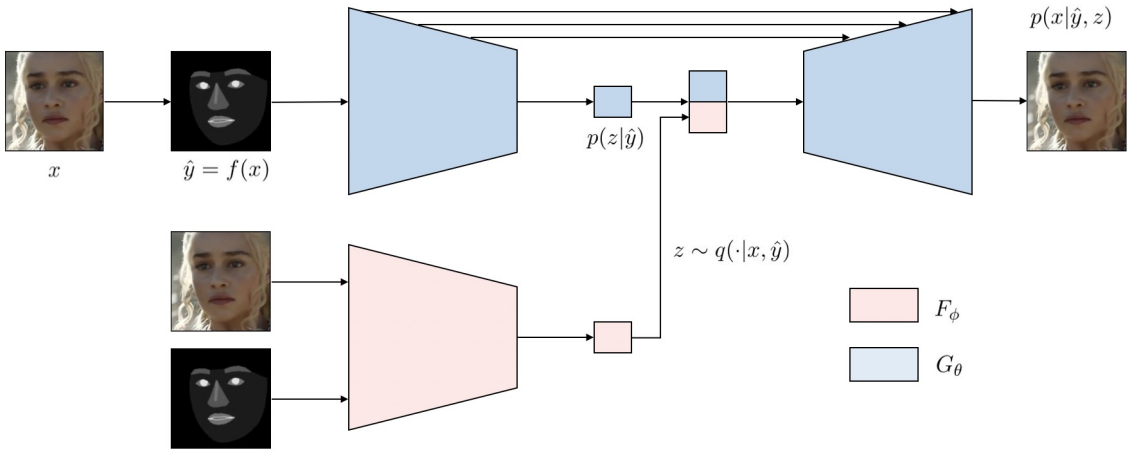


Figure 4.14: **Architecture of VU-Net:** the appearance encoder F_ϕ takes the input image and the shape estimator as inputs, and estimate the mean and variance parameters for the latent appearance distribution. A U-Net takes the shape input to produce shape embedding, which is concatenated with an appearance feature sampled from the latent appearance distribution for image generation.

- **GANimation** performs well on inner face components editing for local regions, as well as preserving identity information, but as it is only conditioned on the target action unit input, it cannot deal with any head pose transformation, and it cannot handle all expressions.

- **VU-Net** is confronted with the obvious identity shift problem and generates blurred inner face. In the original setting of VU-Net, the input shape and appearance information belong to the same object, while the pose guide image and reference image belong to different identities for face reenactment, which brings a gap in identity.

- **gauGAN** has the same identity shift problem, as the original problem setting is also image style transfer, without consideration for the identity gap between the shape input and appearance input in face reenactment problem.

- **PG2** cannot obtain satisfying performance in this face reenactment problem in face appearance preserving due to the paired-training setting as no groudtruth is available, although it can transfer pose well. Another weakness is that PG2 cannot process the reference image and pose guide image separately, but to incorporate them from the beginning, which further degrades the identity preserving performance.

The generated result from the proposed framework without the post-processing of FusionNet can generate high-quality image, but with some local texture details lost, such as wrinkles or mustache. The last column shows that our proposed method can simultaneously modify the expression of the inner face and the whole head pose while still preserving the face identity. See Section 4.7.1 for the quantitative comparison results between the proposed model and single-image methods.

Figure 4.15: **Qualitative comparison with state-of-the-art single-image methods:** our proposed framework generates faces effectively under larger pose changes compared to GANimation [45], with better identity consistency compared to PG2 [36], gauGAN [43] and VU-Net [15].

4.4 Target-specific Method Baseline

Here we compare our model with the state-of-the-art target specific algorithm for face reenactment, ReenactGAN [57]. Instead of directly transferring the facial images, ReenactGAN first maps the input pose guide face to a boundary latent domain. Then a transformer is learned to adapt the boundary of the pose guide face to the target boundary of the reference image. A target-specific decoder will finally generate the reenactment result based on the transformed boundary (see Figure 4.16).

Architecture The overall architecture consists an general encoder, a target-specific decoder and a many-to-one boundary transformer. For the encoder (denoted by ϕ) and target-specific decoder (denoted by ψ), the goal is to learn the suitable ϕ and ψ, such that $\phi : \mathcal{X} \rightarrow \mathcal{B}$ is the function to map the color face image to a latent boundary space, and $\psi : \mathcal{B} \rightarrow \mathcal{X}$ maps the boundary space to a specific person's face, where $\mathcal{X}$ and $\mathcal{B}$ represents the image space and boundary space respectively. The design of the encoder and decoder follows pix2pix [21]. The encoder includes convolutional layers, residual units, and a Hourglass module. The decode is designed in a U-Net fashion. The encoder is shared among

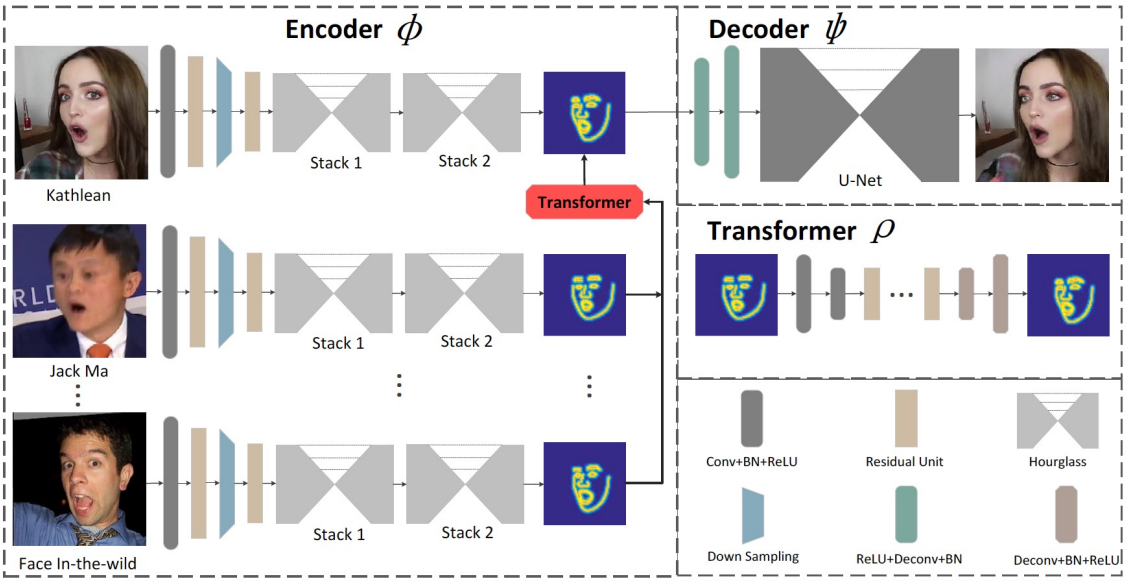


Figure 4.16: **Architecture of ReenactGAN:** an general encoder and the target specific encoder is first trained for photo-realism reconstruction, and a many-to-one target specific boundary transformer is then trained to bridge the gap in the shape latent space between the input reference person and the pose guide image. Figure taken from [57].

all faces, while the decoder has different parameters for different person's input reference images.

However, there is a gap in the shape domain between the input reference person and the pose guide person. A boundary latent representation is introduced to characterize the shape latent space, which eliminates all appearance, background or environment information, but with only the facial boundary information remained as the guidance for structural information. A boundary transformer is implemented to bridge the gap between the face shape of the reference and pose guide image. The boundary transformer is also target-specific, and the boundary transformer for person i is denoted by ρ_i, with an encoder-decoder structure, consisting of residual units, [conv, BatchNorm, ReLU], and [deconv, BatchNorm, ReLU] blocks.

Learning In the target-specific setting, the model can be trained on a video from the target person $\{x_1^T, \ldots, x_M^T\}$, which provides enough appearance information of the target person under different expressions but the same environment. For the training, the encoder and target-specific decoder are first trained jointly without the boundary transformer, and the general loss function can be expressed as:

$$L(\phi, \psi_i, \theta) = L_{GAN}(\phi, \psi_i, \theta) + L_{l1}(\phi, \psi_i) + L_{perceptual}(\phi, \psi_i) \tag{4.17}$$

where ψ_i denotes the decoder for person i, and θ is the discriminator in the adversarial training. $L_{GAN}(\phi, \psi_i)$ is the adversarial loss from GAN, $L_{l1}(\phi, \psi_i)$ is the L1 reconstruction loss between the generated image and the input image since the current encoder and decoder aims to reconstruct image of person i, and the last term $L_{perceptual}(\phi, \psi_i)$ is the perceptual loss which is calculated from the L2 distance between relu2_2 and relu3_3 of the VGG-16 network. By this training stage, a unified encoder which is shared by all faces and a target-specific decoder which captures the specific identity information for person i are obtained.

The second training phase is to train the boundary transformer ρ for each person i. The loss function consists of three parts: a cycle loss, the GAN loss and a shape loss. Let b_i denote the boundary of person i. Since the transformer aims to map an arbitrary person's facial boundary to the reference person's boundary with no paired data for training, a cycle loss is included to sample a transformer ρ_j of a random persons j, and ρ_i is used to reconstruct the corresponding boundary of person i (as shown in Figure 4.17):

$$L_{cycle} = E_{i \neq j} \| \rho_i \cdot \rho_j(b_i) - b_i \| \tag{4.18}$$

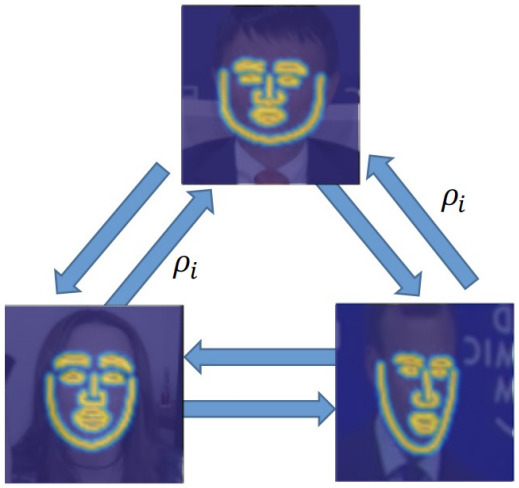


Figure 4.17: **Cycle loss learning of boundary in [57]:** as no paired data is available for boundary transformer learning, a cycle loss is adopted here fore reconstruction. Figure taken from [57].

A different discriminator D_i is trained for each person i to distinguish the reference person's real boundary from the generated boundary, and a vanilla GAN is adopted to calculate the GAN loss L_{GAN}. Besides, a shape constraint L_{shape} is implemented to force the generated boundary to follow the input boundary, i.e., to encourage expression consistency between the input and output of the boundary transformer, by calculating the distance between the input and output boundary after a PCA transformation. Therefore the joint loss for training the boundary transformer is:

$$L(\rho, D) = L_{GAN} + L_{cycle} + L_{shape} \tag{4.19}$$

4.5 Comparison with Target-specific Method

The target-specific method usually can achieve satisfying performance in face reenactment, especially with regard to identity preserving performance, but in practice it has three shortcomings:

- It is usually difficult to collect large amounts of high-quality data of the same person under the same environment but with different expressions.
- It requires expensive computation resources to retrain the target-specific model if you want to reenact a new person.
- It requires expert knowledge to retrain the model, bridging a gap for normal users.

In contrast, our one-shot framework enables users to generate reenactment photos or videos with a single target reference photo input given our trained model, without requirements for extra training nor expert knowledge in this area, allowing normal end-users to experience the reenactment application directly without any extra effort.

Figure 4.18 shows that the target-specific model fails when given a limited number of target data as it requires a large amount of data from the target person to learn the specific identity and appearance information, while our proposed method achieves competitive results taking only one image of each target person.

Figure 4.18: **Qualitative comparison with target-specific methods:** ReenactGAN requires adequate data to train target-specific model for each reference person and fails when the training data decreases significantly. Contrast to that, with a few or even only one reference image, our framework is able to generate competitive results compared with ReenactGAN.

4.6 Visualization Results

Synthesized images A subset of the final visualization results synthesized from the proposed one-shot face reenactment model are shown in Figure 4.19, where the images are generated with the top row images as the reference input, and the left column images as the pose guide input.

Effects of FusionNet To further make comparison between the generated images without and with the proposed FusionNet, Figure 4.20 zooms in some facial regions of the generated images to observe the facial textures in detail such as the eye make-up or mustache. As can be observed, the FusionNet can recover the local facial texture details which are lost during the expression transfer process, as the traditional warping method will only warp the pixels in the image, but will not transform other attributes. A quantitative evaluation for FusionNet in terms of the aforementioned three evaluation metrics will be shown in Section 4.7.1.

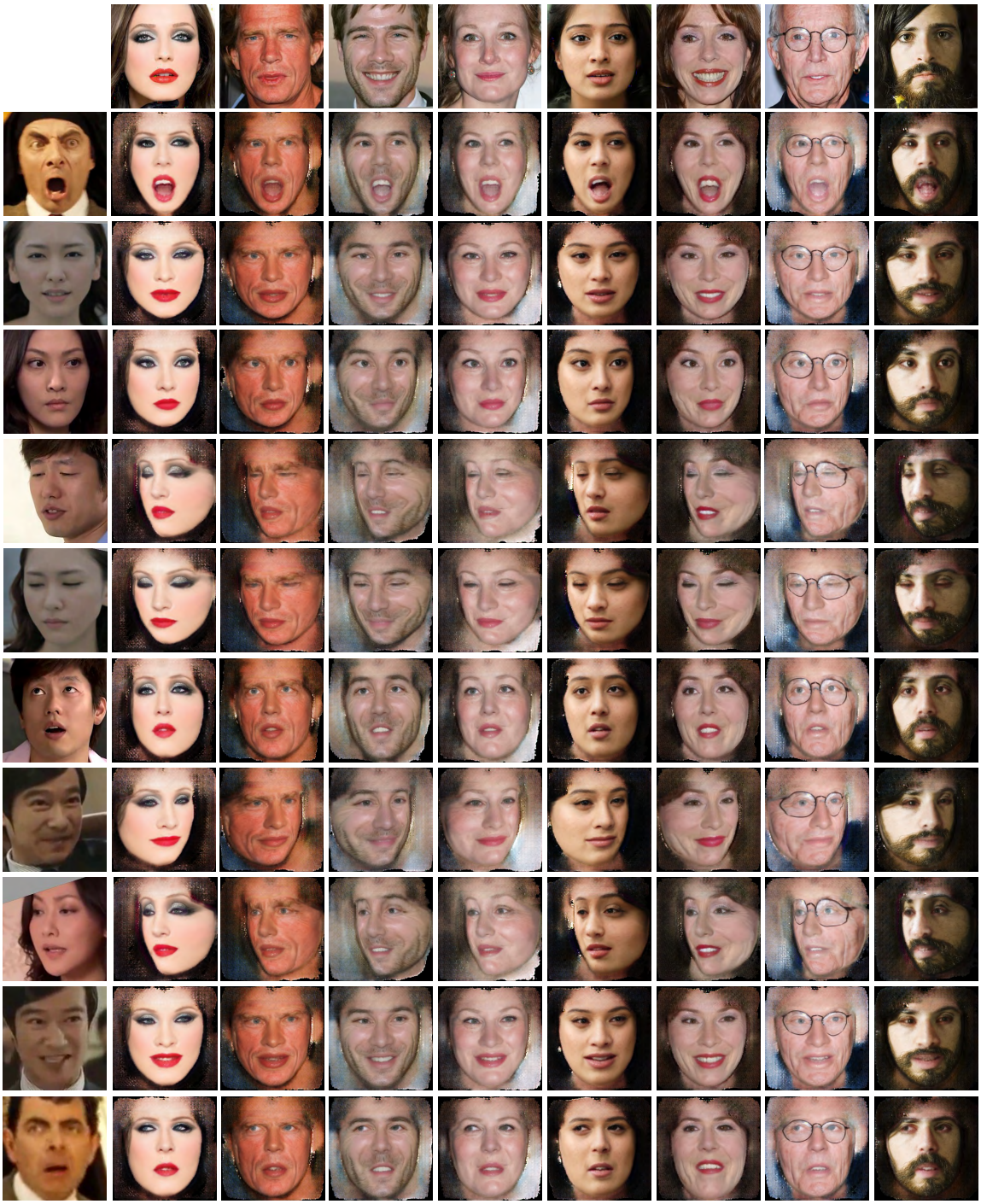


Figure 4.19: **Synthesized images of the proposed model:** with only one reference image input (the top row), our model can reenact under an arbitrary pose guide (the left column). Notice that all result images are generated by one model trained by one-shot data.

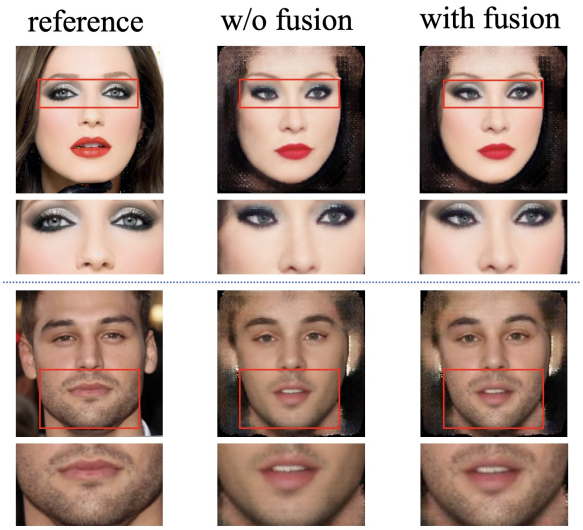


Figure 4.20: **Effects of the proposed FusionNet:** the FusionNet can recover the lost facial texture details (make-up, wrinkles, mustache, etc.), therefore improving the identity consistency between the generated face and the reference input.

4.7 Quantitative Evaluation

4.7.1 Quantitative Comparison with Single-image Methods

We also quantitatively compare our method and the aforementioned single-image baseline methods from three aspects: 1) Expression consistency of inner facial components. 2) The consistency of head pose between the source image and the generated result. 3) The capability of preserving identity, with the three metrics defined earlier, namely the facial AU consistency, head pose consistency, and identity preserving. The quantitative comparison is implemented over three different test datasets as stated in Section 4.1.2: same source (a test split on CelebA-HQ), different source (selected from FFHQ), and in the wild (selected from RAF-DB). We also prepare a baseline that treats reference faces as output, which should have the lowest consistency on facial AU or pose consistency and no identity shifting.

Table 4.1 summarizes all quantitative results; we can observe that:

- The identity preserving ability of our proposed framework overrides the state-of-the-art one-shot methods of other image domains and is competitive with GANimation.
- Our proposed framework performs well in AU and pose consistency while GANimation has frozen pose and cannot handle all expressions.
- Our proposed model has an overall high performance on different sources of data.
- The FusionNet improves identity preserving significantly but with a little drop in facial AU consistency, because the FusionNet focuses on the quality of facial details rather than the expression. Note that the PG2 is trained by paired data which has better supervision in expression, so it has the best AU consistency.

Table 4.1: **Quantitative comparison:** we evaluate each method's consistency of facial action unit (AU), head pose, and the ability in preserving identity. Here we divide the testing data into three groups: 1) same source. 2) cross source (different source but with similar distributions of gender and ethnicity). 3) in the wild. For the reported AU consistency and Id preserving metrics, higher is better, whereas for the pose consistency metric lower is better. All testing samples are driven by the pose guide expressions collected from wild (see Section 4.1.2 for pose guide image sources).

	same source			cross source			in the wild		
Method	AU %	Pose	Id%	AU %	Pose	Id%	AU %	Pose	Id%
Reference	0.00	7.49	-	0	7.77	-	0	7.51	-
GANimation [45] *	18.2	7.49	99.6	17.1	7.77	99.5	13.5	7.50	99.3
VUNet [15]	80.2	3.04	46.1	79.4	3.03	47.7	79.6	3.03	52.2
PG2 [36]	82.7	3.19	52.3	82.6	3.18	53.7	82.3	3.18	51.9
GauGAN [43]	81.1	2.98	79.3	79.72	2.98	75.3	79.7	3.00	78.7
Ours w/o Fusion	80.0	2.79	89.1	80.22	2.84	88.2	80.0	2.91	90.2
Ours	75.1	2.72	98.2	70.9	2.74	98.5	71.1	2.63	98.3

* Different from our training setting, GANimation is supervised by action-unit labels.

4.7.2 Ablation Study

One of the most important part for identity preserving capacity of the proposed one-shot face reenactment model is the multi-scale concatenation of the feature maps from the decoder of appearance auto-encoder F to the semantically adaptive decoder D (indicated by the red arrows in Figure 3.2), as it forces low-level appearance information consistency between the reference image and the generated face, in addition to the high-level identity consistency which is optimized by the identity loss function.

In order to investigate the contribution of the multi-scale concatenation of the appearance feature maps, and the influence of loss weight λ of the auto-encoder reconstruction loss term $L_{app\_reconstruct}$, an ablation study is conducted in terms of the identity preserving metric values, in cases of:

- without the multi-scale concatenation
- different values of λ on the same source test data ($\lambda = 3, 10, 25, 50$)

Table 4.2 provides the ablation study results. We can see that multi-scale concatenation boosts identity preservation performance significantly, and an overly large λ will cause overfitting on the appearance reconstruction on training data, while a small λ cannot provide enough restrictions on appearance information recovery for the generated image. The value of $\lambda = 25$ is chosen in the final training setting.

Table 4.2: **Ablation study:** we evaluate the identity preserving performance under different appearance reconstruction settings on the same source test data.

	No concat	$\lambda = 3$	$\lambda = 10$	$\lambda = 25$	$\lambda = 50$
Id%	77.7	84.9	86.9	**89.1**	85.2

4.7.3 Analysis of k-shot Setting

We also inspect how the one-shot model will perform in a few-shot setting. As there might be only one image for each person taken in one specific environment in the test dataset, we collect an additional

few-shot dataset extracted from several films as the few-shot reference input dataset. There are a total of 32 different identities and each identity has seven images under the same environment but with different expressions and head poses. Then we conduct 3-shot and 5-shot experiments to examine if more data can improve the performance of the propsoed one-shot face reenactment model. The exact method for k-shot setting is described as following: in each k input reference images of the same identity, the nearest neighbour of the pose guide image in the pose space, i.e., the image with the most similar pose with the pose guide image, will be chosen for the synthesis.

Table 4.3 shows the identity preserving performance under few-shot data and our model performs better when more data is available. This is a reasonable result, as more reference inputs for each person can provide more different views of the appearance information, and the model will surely have better synthesis performance with the view which is nearest to the pose guide as the reference input, with no need to guess lots of appearance information.

Table 4.3: **Few-shot experiment:** we evaluate the identity preserving performance under few-shot data.

	one-shot	3-shot	5-shot
Id%	97.2	99.3	**99.4**

Chapter 5

Discussion and Conclusions

5.1 Discussion

Although the general qualitative and quantitative performance of our proposed one-shot face reenactment model combined with the FusionNet are very impressing, especially in pose and expression consistency, the synthesis still faces a problem in identity shifting. To force the identity consistency between the input reference image and the generated face, two methods are adopted in the proposed mdoel from two perspectives:

- Architecture design: the multi-scale appearance feature concatenation between the decoder in the appearance auto-encoder F and the semantically adaptive decoder D.
- Loss function design: the identity loss between the reference and output image calculated by the pretrained face identification network.

However, the reenacted results sometimes share similar facial structure with the pose guide image, due to the reason that the face parsing maps still can maintain a small part of the identity information of the pose guide images, such as the face boundary shape, the size of the facial features, etc., although it has fully got rid of the texture details. And the weakly-supervised training manner cannot fully enable the network to discard these remained identity-related information in the face parsing map, which means the latent space for expression or pose and appearance or identity is not fully disentangled.

As shown in Figure 5.1, the output image may follow the face shape from the pose guide (5.1 (a)), or nose shape (5.1 (b)), or the eye size (5.1 (c)) in cases of the existence of a large gap in identity features between the reference face and pose guide face. Further experiments were conducted later to try to disentangle the latent space via different architectures and training methods, but due to the time restriction, no further significant improvements have been achieved for the proposed one-shot face reenactment.

One-shot face reenactment still remains to be an open and challenging research task, due to its complex requirements in head pose, facial expression and appearance information. We believe that our model has been a pioneering work in this topic, and more progress of this area will be made by researchers in the future.

5.2 Own Contributions

The proposed one-shot face reenactment model has been organized into a paper 'One-shot Face Reenactment', and accepted by BMVC 2019 as a spotlight paper. As it is teamwork, my own contributions are stated as following:

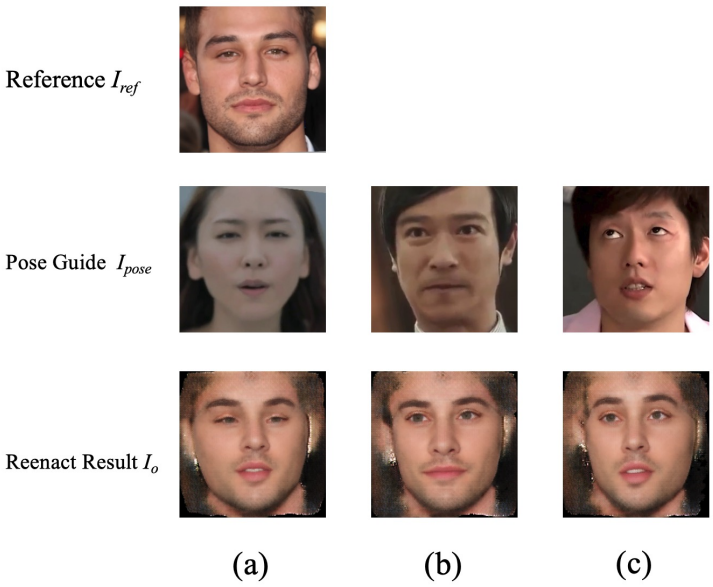


Figure 5.1: **Failure cases:** (a) inconsistent face shape (b) inconsistent nose shape (c) inconsistent eye size.

- Propose the semantically adaptive decoder D, which utilizes the SPADE blocks to incorporate the appearance feature and face parsing maps, to enable better learning performance. Such architecture eliminates the necessity to use a commonly used U-Net-like structure which requires more parameters and results in a slower inference phase.

- Adopt face parsing maps as the latent encoding for shape information. Compared with facial boundary or landmark information, face parsing maps makes the synthesis easier to train as it provides more structural information than other shape representation methods.

- Implement all experiments to achieve the final model with the aforementioned performance in the thesis, including implementing the architecture, tuning parameters, choosing loss functions, ablation study, etc..

- Implement all baselines of single-image method, including the qualitative and quantitative evaluation and comparison between the baselines and our proposed one-shot face reenactment model, as well as the result analysis of the proposed model.

5.3 Conclusions

We have presented a novel one-shot face reenactment model that takes only one image input from the target reference face and is capable of transferring any source face's expression to the reference face. To conclude, we propose the following novel ideas:

- We leverage disentangle learning to model the appearance and the expression space independently and then compose obtained information into reenactment results.

- Specifically, we propose to utilize the face parsing maps as the pose guide information, to enable better learing performance.
- We also introduce a FusionNet to further leverage the generated result and the result from traditional warping method to improve the performance on texture and mustache.
- Our approach is trained in a weakly-supervision manner, with only one target image per subject, and achieves competitive results compared to those using a set of target images, demonstrating the practical merit of this work.

With the aforementioned merits, our one-shot face reenactment does not need paired data, nor multiple images from the same person, but can perform one-shot transformation with impressive performance. Of course, the model still faces problems in identity shifting, due to the remained identity-related information in face parsing maps and weakly-supervision training, which is still an open question for current research in this area, and remains to be further explored in the future.

Bibliography

[1] https://github.com/alyssaq/face_morpher.

[2] https://github.com/deepfakes/faceswap.

[3] https://github.com/shaoanlu/faceswap-gan.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[5] Guha Balakrishnan, Amy Zhao, Adrian V Dalca, Fredo Durand, and John Guttag. Synthesizing images of humans in unseen poses. In *CVPR*, 2018.

[6] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: fine-grained image generation through asymmetric training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2745–2754, 2017.

[7] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Towards open-set identity preserving face synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6713–6722, 2018.

[8] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[9] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. *arXiv preprint arXiv:1808.07371*, 2018.

[10] Huiwen Chang, Jingwan Lu, Fisher Yu, and Adam Finkelstein. Pairedcyclegan: Asymmetric style transfer for applying and removing makeup. In *CVPR*, 2018.

[11] Ying-Cong Chen, Huaijia Lin, Michelle Shu, Ruiyu Li, Xin Tao, Xiaoyong Shen, Yangang Ye, and Jiaya Jia. Facelet-bank for fast portrait manipulation. In *CVPR*, 2018.

[12] Yi-Ting Cheng, Virginia Tzeng, Yu Liang, Chuan-Chang Wang, Bing-Yu Chen, Yung-Yu Chuang, and Ming Ouhyoung. 3d-model-based face replacement in video. In *SIGGRAPH'09: Posters*. ACM, 2009.

[13] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018.

[14] Kevin Dale, Kalyan Sunkavalli, Micah K. Johnson, Daniel Vlasic, Wojciech Matusik, and Hanspeter Pfister. Video face replacement. *ACM Transactions on Graphics (TOG)*, 30(6):1–10, 2011.

[15] Patrick Esser, Ekaterina Sutter, and Björn Ommer. A variational u-net for conditional appearance and shape generation. In *CVPR*, 2018.

[16] Pablo Garrido, Levi Valgaerts, Hamid Sarmadi, Ingmar Steiner, Kiran Varanasi, Patrick Perez, and Christian Theobalt. Vdub: Modifying face video of actors for plausible visual alignment to a dubbed audio track. In *Computer graphics forum*, volume 34, pages 193–204. Wiley Online Library, 2015.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[18] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[19] Qiyang Hu, Attila Szabó, Tiziano Portenier, Paolo Favaro, and Matthias Zwicker. Disentangling factors of variation by mixing them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3399–3407, 2018.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[22] Xiaohan Jin, Ye Qi, and Shangxuan Wu. Cyclegan face-off. *arXiv preprint arXiv:1712.03451*, 2017.

[23] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[24] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[25] Hyeongwoo Kim, Pablo Carrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Transactions on Graphics (TOG)*, 37(4):163, 2018.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[28] Steven M LaValle, Anna Yershova, Max Katsev, and Michael Antonov. Head tracking for the oculus rift. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 187–194. IEEE, 2014.

[29] Shan Li, Weihong Deng, and JunPing Du. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *CVPR*, 2017.

[30] Yining Li, Chen Huang, and Chen Change Loy. Dense intrinsic appearance flow for human pose transfer. In *CVPR*, 2019.

[31] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. *arXiv preprint arXiv:1905.01723*, 2019.

[32] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[33] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[34] Ziwei Liu, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017.

[35] Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. Conditional cyclegan for attribute guided face image generation. *arXiv preprint arXiv:1705.09966*, 2017.

[36] Liqian Ma, Xu Jia, Qianru Sun, Bernt Schiele, Tinne Tuytelaars, and Luc Van Gool. Pose guided person image generation. In *NIPS*, 2017.

[37] Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, and Mario Fritz. Disentangled person image generation. In *CVPR*, 2018.

[38] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[39] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[40] S Mohammad Mavadati, Mohammad H Mahoor, Kevin Bartlett, Philip Trinh, and Jeffrey F Cohn. Disfa: A spontaneous facial action intensity database. *IEEE Transactions on Affective Computing*, 4(2):151–160, 2013.

[41] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[42] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[43] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

[44] Xi Peng, Xiang Yu, Kihyuk Sohn, Dimitris N Metaxas, and Manmohan Chandraker. Reconstruction-based disentanglement for pose-invariant face recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1623–1632, 2017.

[45] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. In *ECCV*, 2018.

[46] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[49] Attila Szabo, Qiyang Hu, Tiziano Portenier, Matthias Zwicker, and Paolo Favaro. Understanding degeneracies and ambiguities in attribute transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 700–714, 2018.

[50] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. *ACM Transactions on Graphics (TOG)*, 34(6):183–1, 2015.

[51] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *CVPR*, 2016.

[52] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *CVPR*, 2017.

[53] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[54] Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popović. Face transfer with multilinear models. *ACM Transactions on Graphics (TOG)*, 24(3):426–433, 2005.

[55] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[56] Olivia Wiles, A Sophia Koepke, and Andrew Zisserman. X2face: A network for controlling face generation using images, audio, and pose codes. In *ECCV*, 2018.

[57] Wayne Wu, Yunxuan Zhang, Cheng Li, Chen Qian, and Chen Change Loy. Reenactgan: Learning to reenact faces via boundary transfer. In *ECCV*, 2018.

[58] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2018.

[59] Taihong Xiao, Jiapeng Hong, and Jinwen Ma. Elegant: Exchanging latent encodings with gan for transferring multiple face attributes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 168–184, 2018.

[60] Runze Xu, Zhiming Zhou, Weinan Zhang, and Yong Yu. Face transfer with generative adversarial network. *arXiv preprint arXiv:1710.06090*, 2017.

[61] Hongyu Yang, Di Huang, Yunhong Wang, and Anil K Jain. Learning face age progression: A pyramid architecture of gans. In *CVPR*, 2018.

[62] Raymond Yeh, Ziwei Liu, Dan B Goldman, and Aseem Agarwala. Semantic facial expression editing using autoencoded flow. *arXiv preprint arXiv:1611.09961*, 2016.

[63] Weidong Yin, Ziwei Liu, and Chen Change Loy. Instance level facial attributes transfer with geometry-aware flow. In *AAAI*, 2019.

[64] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. *arXiv preprint arXiv:1905.08233*, 2019.

[65] Feifei Zhang, Tianzhu Zhang, Qirong Mao, and Changsheng Xu. Joint pose and expression modeling for facial expression recognition. In *CVPR*, 2018.

[66] Gang Zhang, Meina Kan, Shiguang Shan, and Xilin Chen. Generative adversarial network with spatial attention for face attribute editing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 417–432, 2018.

[67] Jiangning Zhang, Xianfang Zeng, Yusu Pan, Yong Liu, Yu Ding, and Changjie Fan. Faceswapnet: Landmark guided many-to-many face reenactment. *arXiv preprint arXiv:1905.11805*, 2019.

[68] Bo Zhao, Xiao Wu, Zhi-Qi Cheng, Hao Liu, Zequn Jie, and Jiashi Feng. Multi-view image generation from a single-view. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 383–391. ACM, 2018.

[69] Hang Zhou, Yu Liu, Ziwei Liu, Ping Luo, and Xiaogang Wang. Talking face generation by adversarially disentangled audio-visual representation. In *AAAI*, 2019.

[70] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.